Centro de Investigación en Matemáticas, A.C.

CIMAT

Random Forests and Autoencoders with Missing Data

T H E S I S

presented in partial fulfillment
of the requirements for the degree of

PhD in Probability and Statistics

by

Irving Gómez Méndez

Advisor:
Dr. Emilien Joly

Guanajuato, Guanajuato,
February 2021

Acknowledgments

I am more than thankful with CIMAT for all the support provided in different ways, which together with the scholarship granted by CONACyT allowed the successful conclusion of this project. Moreover, I would like to thank all the assistant from CIMAT as well as the government of Guanajuato, through the Educafin program, for a fruitful stay at Inria-Lille.

I appreciate all the help, interest and guidance from my advisor Emilien Joly, as well as for all the freedom and trust granted throughout these years. Additionally, I want to take this opportunity to thank Jill-Jênn Vie for always made me feel welcome in France, for sharing your culture with me and for all those enjoyable chats. I would like to thank both of you for keeping the good mood in all circumstances.

I am grateful to all my friends for all your help, support, smiles and laughs: Rocío, Cricelio, Jesús Manuel, Carolina, Laritza, Karla. Special thanks go to Montserrat and Monyrattanak for mantaining a pleasant and warmful environment at home. I would like to warmly thank those friends who has become family, Diana and Enrique. Guys, I thank you all for sharing the spark of friendship with me.

Last but not least, I would like to thank my family; my grandparents: Susy, René, Refugio and Macario; my parents: René and Paty; and my brothers: Eric and René. I am forever grateful for your unconditional support and love.

Abstract

This work addresses the problems of regression with missing data through random forests and the reconstruction of the original data set using autoencoders. Existing algorithms that estimate the regression function using random forests with missing data can be divided in two groups. The first group refers to algorithms that manage missing values directly in the building of recursive trees. The second group is composed by algorithms that attempt to impute the missing values through iterative procedures where the improvement is done through the construction of random forests using the current imputations. After a review on previews approaches, a new method is proposed. The proposal modifies the original CART criterion to overcome the problems faced by this criterion when attempts to handle missing values. Then, under some conditions, we show that this approach generates consistent estimates of the regression function. On the other hand, in a simulation study it is shown that the proposed procedure can obtain similar results to state-of-the-art methods in terms of mean squared error and bias. We discuss some drawbacks of the proposal and give ideas on how to adapt it to handle large amount of missing data. A second part of the work is devoted to the use of autoencoders to reconstruct original observations. This reconstruction is done through the representation in a latent space of the observed part of the input. Denoising autoencoders are introduced as a popular procedure to learn a robust representation of the data in a lower dimension, injecting noise to the original observations. We discuss how the noise injection procedure can help to reconstruct an observation with missing values. After a review of variational inference, a new family of autoencoders known as variational autoencoders is presented as a state-of-the-art generative method which can be adapted to handle missing values. Finally, we develop a lower bound for the evidence and the missing mechanism, present some extensions of these algorithms and briefly discuss how they might be introduce in the field of recommender systems.

Contents

Introduction

In various fields of science, technology and humanities, experts aim at predicting a phenomenon based on past observations or measurements. For instance, meteorologists try to forecast the weather for the next days or weeks from the climatic conditions of the previous days. In medicine, clinical information is collected for diagnosing the condition of patients. In all of these cases, the goal is the prediction of a response variable based on a set of observed predictor variables. When the possible values taken by the response variable belong to a discrete set of values, the prediction of the response from observed predictor variables is known as classification, for example when a patient is diagnosed from a set of possible diseases based on its clinical information. When the possible values of the response belong to a continuous space, then the prediction of its value from observed predictor variables is known as regression, for example when the temperature of the next days is predicted from previous climatic conditions.

Statistical methods for regression and classification have been used for centuries to help researchers and practitioners with these problems of prediction. However, the increasingly speed at which data is generated and the variety of its type has surpassed the limits of standard statistical methods. Fortunately alongside the progress in data generation, new techniques and algorithms from the field of machine learning have been developed as powerful tools for the analysis of complex and large data. Some of these algorithms are the random forests and the neural networks which has been successfully involved in problems of regression and dimensionality reduction. Most of these techniques have been developed considering that all the variables considered in the analysis are available. Although, in practice it is common to deal with data sets that has missing values.

The present work might be divided in two parts. The first one is dedicated to the study and discussion of different techniques based on random forests for regression with missing values, we emphasize and discuss the distinction between two groups. The first group is form by algorithms that attempt to reconstruct the original values using random forests to improve the reconstruction iteratively, sometimes making use of extra structures like the proximity matrix. The second group is composed by algorithms that handle the missing values directly into the composition of the recursive trees rather than trying to impute the missing values. We introduce an alternative algorithm to manage missing data in regression problems, which is achieved through an adaptation of the original random forests algorithm built with the CART criterion. We show through an example the problem presented in the original algorithm which makes it non-viable

in the presence of missing entries. Our research shows the consistency of the proposal when the introduction of missing values is made completely at random and when the regression function can be expressed as an additive model. We discuss some implications of the assumptions presented in this work and give some ideas on how to extend the results under more general conditions. We also develop a simulation study comparing the proposed approach with other algorithms to handle missing values using random forests. The compared techniques include simple baselines, algorithms that iteratively impute the missing values and the Missing Incorporated in Attributes (MIA) which is an approach that manage the missing values directly in the construction of the trees. The simulation study considers the introduction of missing values in distinct variables under seven different mechanisms of missingness, and shows that the proposal can assess similar behavior as state-of-the-art methods. Moreover, study includes the comparison varying the percentage of missing values, discuss some drawbacks and possible ways to solve them so the proposed algorithm could scale efficiently with the amount of data. In our simulation study we also break our CART criterion into four different parts, giving a possible interpretation to these components and showing their behavior under a missing completely at random (MCAR) scenario.

The second part of this work was developed during a stay at Inria-Lille, in collaboration with Dr. Jill-Jênn Vie. It is devoted to the reconstruction of the original data set from important characteristics, which in turn are learned just with the available information in the incomplete data set. This part focuses on the use of a particular family of neural networks known as autoencoders, which were first introduce as a non-linear dimensionality reduction technique and then extended to ensure robust representation of the original inputs into a latent space of lower dimension. The robustness is achieved through the injection of a small amount of noise which forces the autoencoders to learn useful representations. The original inputs might be reconstructed from this representation, yielding a new family of autoencoders known as denoising autoencoders. This technique has been useful when the noise injection is replaced with a procedure that creates "blank" spaces. Another family of autoencoders arrives with the use of variational inference. These autoencoders learn a regular space to represent the data for generative purposes. That is, they can create new unseen elements similar to those presented in the original data set, sampling from the latent space. Members of this family are known as variational autoencorders and represent the state-of-the-art on the use of neural networks in a variety of fields. We deduce a lower bound for the evidence and the mechanism of missingness. Training variational autoencoders with this lower bound allows their use to reconstruct an original input when there are missing values. These models do not only impute the missing values, but they can be used to generate new data with a similar mechanism of missingness, an important goal when we like to shared sensitive data with a third party. Throughout the discussion on autoencoders with missing data we applied them to the MNIST data set to illustrate their employment. Finally, we discuss some extensions of autoencoders especially on their promising use on recommender systems.

Outline of the Thesis

- Chapter 1: This chapter begins with an introduction on statistical learning, describing the most commons scenarios. This work is principally devoted to the so-called supervised statistical learning, although some aspects of the unsupervised statistical learning play an important part too. Hence,a review on the foundations of supervised statistical learning is offered, focusing on the problems of regression and classification, briefly discussing some differences between the parametric and the non-parametric approaches. The random forests algorithm is introduced in this chapter as well as crucial concepts as the CART criterion and data-missing mechanisms. It is discussed how random forests have been used to handle missing values. After this discussion, I propose a new approach based on a modify version of the CART criterion. This new version of the CART criterion enables the creation of recursive trees that handle the missing values directly in their construction. The chapter then moves to the introduction of some tasks on unsupervised statistical learning, with special interest on dimensionality reduction. Artificial neural networks are defined at this point, discussing some of their applications. After this preamble on neural networks and dimensionality reduction, autoencoders are presented as a family of neural networks that allows the reconstruction of observations from a representation in a space of lower dimension. The knowledge of the necessary concepts and tools will help to comprehend the underlying concepts behind the procedures considered in this work and it will be useful in designing new algorithms to face diverse problems.

- Chapter 2: I present the results of a simulation study in a regression problem with missing values, comparing different methods based on random forests. The study considers seven different mechanisms of missingness similar to those presented in Rieger, Hothorn, and Strobl (2010), and the performance of the methods is measure in terms of their mean squared error. Furthermore, I also compare the different methods in terms of their bias, giving new insights on the behavior of the methods widely ignored in the previous literature. Then, I vary the percentage of the missing values for a wide range of values, from a low number of observations with missing values to data sets where up to 95% of the observations present some missingness. These studies show that the proposed algorithm in Chapter 1 can acquire similar performance to state-of-the-art procedures or even surpass them for data sets with large percentage of missing values. Important computational drawbacks on the proposal are discussed as well as some ideas to overcome these issues. Due to a lack of packages to manage missing values, I programmed the proposed approach as well as the well-known MIA procedure in `R` without relying on previous packages. At the end of the chapter I present a decomposition of the modified CART criterion, giving a possible interpretation to the components and studying them in an MCAR scenario.

- Chapter 3: An overview on previous works that studied the consistency of random forests is offered in this chapter. After this overview I show how the recursive trees built with the modified CART criterion presented in Chapter 1 can be consistent under specific assumptions. This result considers an additive model for the regression function, as was done by Scornet, Biau, and Vert (2015), and missing values introduced accordingly to an MCAR procedure. The complete proof is presented, dividing it in three main parts. The first part is devoted to the study of the theoretical version of the modified CART criterion, and showing that low values

of this version implies minimum variation on the regression function. I present through an example how this implication is not hold when the assumption of an additive regression function is not preserve. In the second part, the empirical modified CART criterion is studied showing that it tends to zero in probability. The third part combine the two first results, showing that asymptotically the regression function has no variation on the final cells constructed empirically. Finally, the consistency is achieved as a consequence of the non-variation of the regression function on the final cells. I briefly discuss the assumptions considered and possible extensions of the results considering more general conditions. Up to my knowledge, this is one of the first results showing the consistency of regression estimators with missing values.

- Chapter 4: Different autoencoders are presented in this chapter, as neural networks capable to learn a reliable space to represent the data and to reconstruct the original observations from this low-dimensional representation. Denoising autoencoders (DAEs) are introduced as an improvement of the original autoencoders to generate more robust representations. With minimal changes, this new family of autoencoders could be used to impute missing value. Hence, a wide discussion on the use of DAEs for imputed purposes is given. Then, a review on variational inference is presented with a new family of autoencoders known as variational autoencoders (VAEs). Originally created for generative purposes, they can also be used to reconstruct an input with missing values. I deduce a lower bound for the evidence and the mechanism of missingness, which can be used to train VAEs. I discuss how this lower bound enables the VAEs to reconstruct the original observations as well as learning the mechanism of missingness. Finally, I briefly discuss some extensions for their implementation in recommender systems. Throughout this chapter I use an example based on the well-known MNIST data set to illustrate the different ideas and methods introduced.

1

Statistical Learning with Missing Data

When scientists learn about nature, the environment can be thought as a passive agent - apples drop, stars shine, and the rain falls without regard the needs of the scientists. We model such learning scenarios by postulating that data is generated by some random process. The task of statistical models is to process such randomly generated examples toward drawing conclusions for the phenomenon from which these examples are picked. Statistical learning uses the strengths and special abilities of computers to perform tasks that fall way beyond human capabilities. For example, the ability to scan and process huge databases allows the statistical learning program to detect meaningful patterns that are outside the scope of the scientists' perception and may have been missed by the human observer. On the other hand, machine learning can be defined as the study of systems that can learn from data without being explicitly programmed, and a computer program is said to learn from data with respect to some task if its performance at that task improves with data (Louppe, 2014; Mitchell et al., 2011). This description highlights the close relationship between machine learning and statistics, and constitutes the basic building block of the statistical learning. As commented by Barcenas (2020), this process typically has two phases:

1. Training phase: In this phase, a model is constructed from a training data set, where usually the response variable is known along with the predictor variables (labeled data), and both of them are used to improve the performance of the model. Hence, the model is trained minimizing some loss function that measures the difference between the prediction made by the model and the true value of the response variable.

2. Testing phase: This phase is characterized by the use of a testing data set which has not made available in the training phase. During the testing phase the constructed model must predict the response variable of the observations in the testing data set. These predictions are then compared with the true value of the response to measure the performance of the algorithm.

We can distinguish different learning scenarios based on the types of data, the order and the method by which the training data and the testing data are received, as well as their use to evaluate the learning algorithms. We briefly describe some common machine learning scenarios, based on the classification made by Mohri, Rostamizadeh, and Talwalkar (2018) and Shalev-Shwartz and Ben-David (2014):

- Supervised learning: It describes a scenario in which the training data set contains labeled observations. The model is then train to predict the label based

on the predictor variables. In such cases we can think of the environment as a teacher that "supervises" the model by providing that extra information (the response variable). This is the most common scenario associated with classification, regression, and ranking problems.

- Unsupervised learning: In this case the model exclusively receives unlabeled training data, that is, observations without the value for the response variable, or even without the existence of such label. The model processes the input data with the goal of coming up with some summary, or compressed version of that data. Since there is an unclear distinction between training and testing data, it can be difficult to evaluate the performance of the model. Clustering and dimensionality reduction are examples of unsupervised learning problems.

- Semi-supervised learning: The model receives a training sample consisting of both, data with the response variable (labeled data) and without it (unlabeled data), and makes predictions for all unseen points. Semi-supervised learning is common in settings where unlabeled data is easily accessible but labels are expensive to obtain. Various types of problems arising in applications, including classification, regression, or ranking tasks, can be framed as instances of semi-supervised learning. The hope is that the distribution of unlabeled data accessible to the model can help it achieve a better performance than in the supervised setting.

- Online learning: In contrast with the previous scenarios, the online scenario involves multiple rounds where training and testing phases are intermixed. In this case the model has to respond online, throughout the learning process, instead of engage the acquired knowledge only after having a chance to process large amount of data. At each round, the model receives a training point without the value of the response variable, makes a prediction, receives the true value, and incurs a loss. The objective in the online setting is to minimize the cumulative loss over all rounds or to minimize the regret, that is the difference of the cumulative loss incurred and that of the best model in hindsight. The model becomes an expert over time, but might have made costly mistakes in the process. In contrast, in the previous scenarios the model has large amount of training data to play with before having to output conclusions.

- Reinforcement learning: The training and testing phases are also intermixed in reinforcement learning. To collect information, the model actively interacts with the environment and in some cases affects it, and receives an immediate reward for each action. The objective of the model is to maximize the reward over a course of actions and iterations with the environment. Hence, the model is faced with the exploration versus exploitation dilemma, since it must choose between exploring unknown actions to gain more information versus exploiting the information already collected.

A learner model that interacts with the environment at training time, say, by posing queries or performing experiments is called an active learner, while a passive learner only observes the information provided without influencing or directing it. Thus, active learning refers to a model that adaptively or interactively collects training observations. The goal in this case is to achieve a performance comparable to the standard supervised learning scenario (or passive learning scenario), but with fewer labeled observations. Active learning is often used in applications where getting the response

variable is expensive, for example computational biology applications. The previous learning scenarios relies somewhere in the middle between the active-passive spectrum.

In this work we discuss only a subset of the possible learning paradigms. Our main focus is on supervised statistical learning with passive learner models, particular interest is on regression with missing data and the reconstruction of the original observations. We also discuss briefly unsupervised learning, in particular dimensionality reduction.

1.1 Regression and Classification

In supervised statistical learning we have an outcome measurement $Y \in \mathcal{Y}$ that we wish to predict based on a set of input variables $\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(p)}$ where $\mathbf{X}^{(j)} \in \mathcal{X}^{(j)}$ for $j = 1, \ldots, p$. Together, the input variables $(\mathbf{X}^{(1)}, \ldots, \mathbf{X}^{(p)})$ form a p dimensional input vector $\mathbf{X}$ taking its values in $\mathcal{X} = \mathcal{X}^{(1)} \times \cdots \times \mathcal{X}^{(p)}$.

The input space $\mathcal{X}$ and the output space $\mathcal{Y}$ are assumed, by definition, to respectively contain all possible input vectors and all possible output values. The input variables are sometimes known as features and the output variable as target. If Y is a continuous variable then the learning task is a regression problem.

Formally, the aim is to find a function $f : \mathcal{X} \to \mathcal{Y}$ that minimizes

$$\mathbb{E}_{\mathbf{X},Y} \left[\mathcal{L}(f(\mathbf{X}), Y) \right]$$

where $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ is a loss function that penalizes errors in the prediction. By far, the most common loss function for regression problems is the squared error loss $\mathcal{L}(f(\mathbf{X}), Y) = (f(\mathbf{X}) - Y)^2$, and the function m which satisfies

$$m = \underset{f:\mathcal{X}\to\mathcal{Y}}{\arg\min}\, \mathbb{E}_{\mathbf{X},Y} \left[(f(\mathbf{X}) - Y)^2 \right]$$

is given by the regression function $m(\mathbf{X}) = \mathbb{E}[Y|\mathbf{X}]$.

If Y takes its values from a finite set without an order, we say that it is a categorical variable and the learning task is a classification problem. In this kind of problems it is usual to consider the 0-1 loss function $\mathcal{L}(f(\mathbf{X}, Y)) = \mathbb{1}_{f(\mathbf{X})\neq Y}$. Let $\mathcal{Y} = \{c_1, \ldots, c_J\}$, the function m which satisfies

$$m = \underset{f:\mathcal{X}\to\mathcal{Y}}{\arg\min}\, \mathbb{E}_{\mathbf{X},Y} \left[\mathbb{1}_{f(\mathbf{X})\neq Y} \right]$$

is called the Bayes classifier, given by $m(\mathbf{X}) = \arg\max_{y\in\{c_1,\ldots,c_J\}} \mathbb{P}\left[Y = y|\mathbf{X}\right]$.

For practical problems, the distribution of $(\mathbf{X}, Y)$ is unknown and hence, the regression function or the Bayes classifier are unknown as well. However, in our framework, we have access to a training data set $\mathcal{D}_n = (\mathbf{X}_i, Y_i)_{i=1,\ldots,n}$ where the collected data has the same distribution than $(\mathbf{X}, Y)$. The goal is to use the data set $\mathcal{D}_n$ to construct a learning model, also called learner or predictor, $m_n : \mathcal{X} \to \mathcal{Y}$ which estimates the function m, and enables us to predict the outcome for new unseen objects.

At least two distinct strategies can be used to solve the estimation problem, the parametric and the non-parametric approaches. In a nutshell, the parametric estimation uses a model belonging to a set of functions $\mathcal{F}_\Theta$ determined by a finite number of parameters $\theta \in \Theta$, then the estimation is made through the inference of this set of parameters that minimize the empirical risk,

$$m_n \equiv m_n(\cdot, \hat{\theta}) = \underset{f_\theta\in\mathcal{F}_\Theta}{\arg\min}\, \mathbb{E}_{\mathcal{D}_n}[\mathcal{L}(f_\theta(\mathbf{X}), Y)]$$

where

$$\hat{\theta} = \arg\min_{\theta \in \Theta} \mathbb{E}_{\mathcal{D}_n}[\mathcal{L}(f_\theta(\mathbf{X}), Y)]$$

Parametric estimates usually depend only on a few parameters, therefore they are suitable even for small sample sizes n. Furthermore, they are often easy to interpret, for example in a linear model (when m is a linear function) the absolute value of the coefficients indicates how much influence the components of $\mathbf{X}$ has on the value of Y, and its sign describes the nature of this influence (increasing or decreasing the value of Y). However, as pointed out by Györfi et al. (2002) parametric estimates have a rigid structure that can make them to produce large errors for every sample size if the true underlying regression function cannot be well approximated by the parametric structure.

As commented by Müller and Quintana (2004) to relax the parametric assumptions and allow greater modeling flexibility and robustness against mis-specification of a parametric statistical model we consider models that cannot be indexed by a finite dimensional parameter. This kind of models are known as non-parametric models. More formally, the non-parametric estimation uses a model belonging to a set of functions $\mathcal{F}_n$ not determined by a finite set of parameters, whose complexity is allowed to increase with n. From a practical point of view, this algorithms are popular since they can be implemented even with little or no context of the problem and without requiring to minimize a certain loss function. However, considerable effort have been put to improve these algorithm and to find optimal values for hyper-parameters (see e.g. Devroye and Lugosi (2012), Tsybakov (2008), and Efromovich (2008)). Furthermore, it has been proved that some non-parametric algorithms actually minimize certain loss function (see Györfi et al. (2002)), so they can be expressed as

$$m_n = \arg\min_{f \in \mathcal{F}_n} \mathbb{E}_{\mathcal{D}_n}[\mathcal{L}(f(\mathbf{X}), Y)]$$

A desirable for an estimate is that, as the sample size grows, it should converge to the estimated quantity, that is., the error of the estimate should converge to zero for a sample size tending to infinity. Estimates which have this property are called consistent. A sequence of regression function estimates $\{m_n\}$ is called weakly consistent for a certain distribution of $(\mathbf{X}, Y)$, if

$$\lim_{n \to \infty} \mathbb{E}_{\mathbf{X}, Y | \mathcal{D}_n}[m_n(\mathbf{X}) - m(\mathbf{X})]^2 = 0$$

It may be that a regression function estimate is consistent for a certain class of distributions of $(\mathbf{X}, Y)$, but not consistent for others. It is clearly desirable to have estimates that are consistent for all the distributions. A sequence of regression function estimates $\{m_n\}$ is called weakly universally consistent if it is weakly consistent for all distributions of $(\mathbf{X}, Y)$ with $\mathbb{E}[Y]^2 < \infty$.

1.2 Random Forests

1.2.1 Introduction to Random Forests

Introduced by Breiman (2001), the random forest algorithm is named after the general technique that consists in building and exploiting the objects called decision trees. These objects are rooted trees in the mathematical sense and are, most of the time, binary trees. Each path starting from the root corresponds to a decision (sometimes called

prediction), and each branch represents a region of the input space. Moving along the path of the decision tree corresponds to a choice of one of the possible regions. Decision trees are conceptually simple yet powerful and attractive in practice for several reasons:

- They can model arbitrarily complex relations between the input and the output space.
- They handle categorical or numerical variables, or a mix of both.
- They can be used in regression or supervised classification problems.
- They are easy interpretable, even for non-statisticians.

To construct the partition of the input space, some decision trees are built in a recursive way, therefore they are called recursive trees. The root of the tree is the whole input space $\mathcal{X}$ which is split into disjoint regions. Then each region is split and this process continues until some stopping rule is applied (see Figure 1.1). At each step of the tree construction, a partition is performed over a cell (or equivalently its corresponding node) maximizing some split criterion.

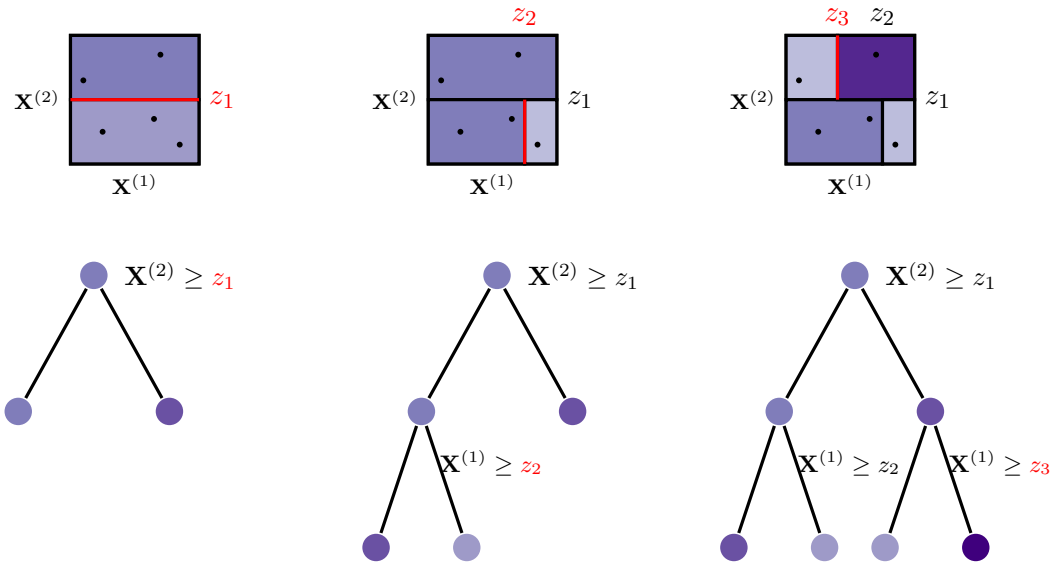

Figure 1.1: The construction of the trees is done in a recursively. At each step of the tree construction a partition is performed over a cell (or equivalently its corresponding node) maximizing some split criterion.

The random forest algorithm is made of a set of $M(> 1)$ randomized (in its construction) recursive trees that are later aggregated all together (with a simple mean idea for regression problems). Since a random forest aggregates the information of many different decision trees in a global predictor, it ends to be more stable (and then more informative) than each specific tree. Randomization is introduced in two different parts of the tree construction. The first source of randomness is through the bootstrap-aggregation technique (Breiman, 1996) -commonly called bagging or subsampling step- which consists in selecting a_n observations randomly with (or without) replacement from the training data set $\mathcal{D}_n$, prior to the construction of each tree. Only these a_n observations are taken into account in the tree construction. The second source of randomness is introduced during the split of a node, this split is performed by maximizing

the split criterion over `mtry` input variables chosen uniformly at random among the original ones. The tree construction is stopped when each final node contains less or equal than `nodesize` points or when the tree has t_n final nodes. Hence, the parameters of this algorithm are:

- $M > 1$, which is the number of trees in the forest.
- $a_n \in \{1, \dots, n\}$, which is the number of observations in each tree.
- `mtry` $\in \{1, \dots, p\}$, which is the number of directions (features) chosen, candidate to be split. We denote by $\mathcal{M}_{try}$ the features selected in each step.
- `nodesize` $\in \{1, \dots, a_n\}$, which is the maximum number of observations for a node to be a final cell.
- Instead of `nodesize` we can use the parameter $t_n \in \{1, \dots, a_n\}$, which is the number of leaves (final nodes) in each tree.

As mentioned by Scornet, Biau, and Vert (2015) this algorithm or some modifications have been successfully involved in various practical problems, however little is known about the mathematical properties of the method which has lead the community to underline a "gap" between theory and practice. Part of this gap can be explained by the bagging mechanism and the splitting criterion. Each of these processes introduces a source of randomness into the construction of the trees which makes the random forest algorithm very challenging to be studied in its full generality. One way to examine the theoretical justifications of random forests is through simplified versions of the original procedure. This is often done by simply ignoring the bagging step and/or by replacing the splitting criterion with a more elementary protocol (see, for example Breiman (2004)). However, in recent years, important theoretical studies have been performed to analyze more elaborated models (e.g. Biau, Devroye, and Lugosi (2008), Ishwaran and Kogalur (2010), Biau (2012), Genuer (2012), and Zhu, Zeng, and Kosorok (2015)). This non-parametric technique has various computational benefits over many other approaches, the small number of hyper-parameters to be tuned in the random forests algorithm makes it fast and simple, and the bagging technique allows a straightforward parallelization which has made random forests one of the most popular tools for handling data sets of large size.

The randomization of the trees (independent from the original source of randomness in the sample $\mathcal{D}_n$) is represented in a symbolic random variable Θ. To each tree – randomized with the random variable Θ_k – is associated a predicted value at a query point $\mathbf{x}$, denoted by $m_n(\mathbf{x}; \Theta_k)$. The different trees are constructed by the same procedure but with independent randomization so that the random variables $\Theta_1, \dots, \Theta_M$ are i.i.d. with common law Θ. The nature and dimension of Θ depends on its use in the tree construction. In our choice of construction rules, Θ consists of the observations selected for the tree and the candidate variables to split at each step.

At the end of the tree construction, a partition of the space $\mathcal{X}$ is returned in a form of a collection of cells $(A_{n,j})_{j \geq 1}$. Each of these cells corresponds to a leaf of the tree and they are called final cells to emphasize their difference with the cells that we consider during the construction of the tree. In order to perform the estimation, only these final cells are necessary. Finally, the kth tree estimate is defined as

$$m_n(\mathbf{x}; \Theta_k) = \sum_{i \in \mathcal{I}_{n,\Theta_k}} \frac{Y_i \mathbb{1}_{\mathbf{X}_i \in A_n(\mathbf{x}; \Theta_k)}}{N(A_n(\mathbf{x}; \Theta_k))}$$

where $\mathcal{I}_{n,\Theta_k}$ is the set of the a_n observations selected prior to the construction of the kth tree, $A_n(\mathbf{x};\Theta_k)$ is the unique final cell that contains $\mathbf{x}$, and $N(A_n(\mathbf{x};\Theta_k))$ is the number of observations which belong to the cell $A_n(\mathbf{x};\Theta_k)$. The aggregation of the trees forms the finite random forest estimate given by

$$m_{M,n}(\mathbf{x};\Theta_1,\ldots,\Theta_M) = \frac{1}{M}\sum_{k=1}^{M} m_n(\mathbf{x};\Theta_k).$$

It is known from the work of Breiman (2001) that the random forest estimate does not overfit when M tends to infinity. This makes the parameter M only restricted by the computational power.

1.2.2 Split Criteria

Different split criteria have been proposed depending on the statistical problem and the nature of the input space. For regression purposes, the most commonly used is the CART criterion (Breiman et al., 1984). For supervised classification, the split criterion usually takes the form of an impurity function to be minimize, like the misclassification error, the Gini index (Gini, 1912) or the criteria known as ID3 and C4.5 (Quinlan, 1986; Quinlan, 1993) which minimize the Shannon entropy (Shannon, 1948) and replace binary splits on categorical variables with multiway splits.

To define the impurity functions for the classification tasks, assume that $\mathcal{Y} = \{c_1,\ldots,c_J\}$, fix a cell A of the tree and denote by $N(A)$ the number of observations belonging to that cell, let be

$$p_{c_j} = \frac{1}{N(A)}\sum_{i=1}^{n} \mathbb{1}_{Y_i=c_j,\mathbf{X}_i\in A}$$

the proportion of observations of the class c_j in node A. We classify the observations in cell A to class $\hat{c}_A = \arg\max_{c_j} p_{c_j}$, the majority class in the cell. The previous impurity functions are then defined as

$$\begin{aligned}
&\text{Missclassification error:} && \frac{1}{N(A)}\sum_{i=1}^{n} \mathbb{1}_{Y_i\neq\hat{c}_A,\mathbf{X}_i\in A} = 1 - p_{\hat{c}_A}\\
&\text{Gini index:} && \sum_{j=1}^{J} p_{c_j}(1-p_{c_j})\\
&\text{Shannon entropy:} && -\sum_{j=1}^{J} p_{c_j}\log_2 p_{c_j}
\end{aligned}$$

For two classes, if p is the proportion in the second class, these three measures are $1-\max\{p,1-p\}$, $2p(1-p)$ and $-p\log_2(p) - [(1-p)\log_2(1-p)]$, which are shown in Figure 1.2. All three are similar, but Shannon entropy and Gini index are differentiable, and hence more amenable to numerical optimization. In addition, Shannon entropy and Gini index are more sensitive to changes in the node probabilities than the misclassification rate. However, while they are robust and reliable impurity functions they are not exempt of defects, like the end-cut preference (Morgan and Messenger, 1973; Breiman et al., 1984), that is the tendency to favor unbalanced splits (in the number of points belonging to each child node) in which p is near 1 or zero, resulting in deep and uninterpretable trees. Furthermore, as some authors have pointed out (Quinlan, 1986; Strobl,

Boulesteix, and Augustin, 2007), they have the propensity of preferring splits based on input variables with many outcomes.

Facing this problem, Hothorn, Hornik, and Zeileis (2006) introduce conditional inference trees. Splits are performed in two steps. In the first step the relation of a variable to the response in assessed by permutation tests based on a theoretical conditional inference developed by Strasser and Weber (1999). After the strongest relation is found by minimal p-value of the permutation tests it is checked if the significance to a certain level is still present after adjustment for multiple testing. Finally, in the second step the best cutpoint is determined for the variable chosen. The growth of the tree stops when no further significant relations are found.

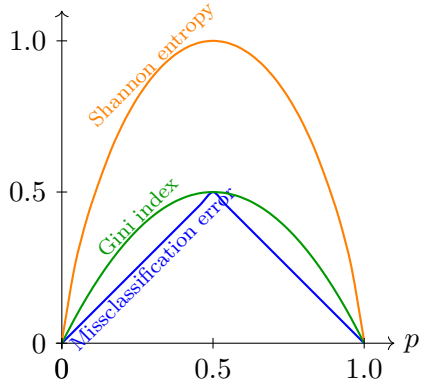

Figure 1.2: Node impurity measures for two-class classification, as a function of the proportion p in the second class.

1.2.3 Formal Definitions for the CART Criterion

The present work focuses on the CART split criterion, although the ideas can be generalized to other criteria. We first introduce some important notations.

- $A = \prod_{h=1}^{p}[a^{(h)}, b^{(h)}]$, denotes a general node (or cell), $a^{(h)}, b^{(h)} \in \mathcal{X}^{(h)}$ for all $h = 1, \ldots, p$.
- $N(A)$ holds for the number of points in A.
- The notation $d = (h, z)$ denotes a cut in A, where

  h is a direction, $h \in \{1, \ldots, p\}$, and

  z is the position of the cut in the hth direction, between the limits of A, that is $a^{(h)} < z < b^{(h)}$.
- $\mathcal{C}_A$ is the set of all possible cuts in the node A. It means that h do belong to the set $\mathcal{M}_{try}$ and that for any chosen h, z lies between the bounds of the cell A in that specific direction.
- A cell A is split into two cells denoted as $A_L = \{\mathbf{x} \in A : \mathbf{x}^{(h)} < z\}$ and $A_R = \{\mathbf{x} \in A : \mathbf{x}^{(h)} \geq z\}$.
- $\bar{Y}_A$ (respectively $\bar{Y}_{A_L}$, $\bar{Y}_{A_R}$) is the empirical mean of the response variable Y_i for the indexes such that $\mathbf{X}_i$ belongs to the cell A (respectively A_L, A_R).

Then, the empirical version of the CART split criterion for a generic cell A and the full sample $\mathcal{D}_n$ is defined as

Definition 1 Empirical CART Criterion

$$\begin{aligned} L_n(A,d) =& \frac{1}{N(A)} \sum_{i=1}^{n} \left(Y_i - \bar{Y}_A\right)^2 \mathbb{1}_{\mathbf{X}_i \in A} \\ & - \frac{1}{N(A)} \sum_{i=1}^{n} \left(Y_i - \bar{Y}_{A_L} \mathbb{1}_{\mathbf{X}_i^{(h)} < z} - \bar{Y}_{A_R} \mathbb{1}_{\mathbf{X}_i^{(h)} \geq z}\right)^2 \mathbb{1}_{\mathbf{X}_i \in A} \end{aligned}$$

with the convention $0/0 = 0$.

Intuitively, the CART criterion compares the empirical variance inside the cell A with the sum of the two empirical variances of the sub-cells A_L and A_R. At each step of the creation of the random tree, the current cell A is selected and split by choosing the best empirical cut $\widehat{d} = (\widehat{h}, \widehat{z})$ so that $L_n(A,d)$ is maximal over the set $\mathcal{C}_A$, that is

Definition 2 Best Empirical Cut

$$\widehat{d} = (\widehat{h}, \widehat{z}) \in \underset{d \in \mathcal{C}_A}{\arg\max}\, L_n(A,d)$$

The theoretical counterpart of the empirical CART criterion is given by

$$\begin{aligned} L^\star(A,d) =& \mathbb{V}[Y|\mathbf{X} \in A] - \mathbb{V}[Y|\mathbf{X}^{(h)} < z, \mathbf{X} \in A]\mathbb{P}[\mathbf{X}^{(h)} < z|\mathbf{X} \in A] \\ & - \mathbb{V}[Y|\mathbf{X}^{(h)} \geq z, \mathbf{X} \in A]\mathbb{P}[\mathbf{X}^{(h)} \geq z|\mathbf{X} \in A], \end{aligned}$$

analogously to the empirical case, we define the best theoretical cut $d^\star = (h^\star, z^\star)$ in A as

$$d^\star = (h^\star, z^\star) \in \underset{d \in \mathcal{C}_A}{\arg\max}\, L^\star(A,d)$$

By the classical strong law of large numbers, $L_n(A,d)$ converges almost surely to $L^\star(A,d)$ as n tends to infinity, for all cuts $d \in \mathcal{C}_A$. This fact leads to the interpretation that the chosen cuts at each step of the tree construction tend to decrease the variability of the sets of data points corresponding to the child nodes. This implies that the empirical mean in each cell tends to stabilize around a value which corresponds to the conditional mean for the cell.

Applying basic algebra, it is easy to get an equivalent expression for the empirical CART criterion, given by

$$L_n(A,d) = \frac{N(A_L)N(A_R)}{N(A)N(A)} \left(\bar{Y}_{A_L} - \bar{Y}_{A_R}\right)^2 \tag{1.1}$$

Thus, from Equation (1.1), we can get an alternative expression to the theoretical CART criterion, given by

$$\begin{aligned} L^\star(A,d) =& \mathbb{P}[\mathbf{X}^{(h)} < z|\mathbf{X} \in A]\mathbb{P}[\mathbf{X}^{(h)} \geq z|\mathbf{X} \in A] \\ & \times \Big(\mathbb{E}[Y|\mathbf{X} < z, \mathbf{X} \in A] - \mathbb{E}[Y|\mathbf{X} \geq z, \mathbf{X} \in A]\Big)^2 \end{aligned}$$

1.3 Neural Networks

Neural networks were developed soon after the advent of computers in the fifties and sixties with an initial excitement about the prospects of artificial intelligence. However, after the initial euphoria, there was a period of disappointment in which the computationally intensive nature of neural networks was seen as an impediment to their usability. Eventually greater data availability and increasing computational power lead to increased successes of neural networks, and this area was reborn under the new label of "deep learning".

Neural networks contain computation units referred to as neurons as an attempt to simulate the human nervous system for machine learning tasks. The computational units are connected to one another through weights, which serve the same role as the strengths of synaptic connections in biological organisms, these weights affect the function computed at that unit. An artificial neural network computes a function of the inputs by propagating the computed values from the input neurons to the output neurons and using the weights as intermediate parameters. Learning occurs by changing the weights connecting the neurons.

These models gain their power by putting together many such basic units, and learning their weights jointly in order to minimize a loss function. From this point of view, a neural network can be viewed as a computational graph of elementary units in which greater power is gained by connecting them in particular way. The training data provides feedback to the correctness of the weights in the neural network depending on how well the predicted output approximates the real target variable in the training data. The goal of changing the weights is to modify the computed function to correct the predictions in future iterations. By successively adjusting the weights between neurons over many input-output pairs, the function computed by the neural network is refined over time so that it provides more accurate predictions. By combining multiple units, we increase the power of the model to learn more complicated functions.

A feedforward network defines a mapping $m_n(\cdot, \theta)$ and learns the value of the weights θ that result in the best function approximation. These models are called feedforward because information flows through the function being evaluated from $\mathbf{X}$, through the intermediate computations used to define m_n, and finally to the output Y. There are no feedback connections in which outputs of the model are fed back into itself. Hence, these models are represented by composing together many different functions and they are associated with a directed acyclic graph describing how the functions are composed. For example, we might have three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $f(\mathbf{X}) = f^{(3)}(f^{(2)}(f^{(1)}(\mathbf{X})))$. In this case, $f^{(1)}$ is called the first layer of the network, $f^{(2)}$ is called the second layer, and so on. The overall length of the chain gives the depth of the model. The learning algorithm must decide how to use those layers to produce the desired output. Figure 1.3 shows a representation of a feedforward neural network with 3 layers. According with Goodfellow et al. (2016) these chain structures are the most commonly used for neural networks. When feedforward neural networks are extended to include feedback connections, they are called recurrent neural networks.

Neural networks have been successfully introduced in diverse problems of machine learning. Natural Language Processing (NPL), in a nutshell, is the ability for a computer/system to truly understand human language and process it in the same way that a human does. Speech recognition, dialog systems, information retrieval, question answering, and machine translation are some applications of NLP. Nowadays neural networks and deep learning have become standard methods applied to practically all NLP tasks. Goyal, Pandey, and Jain (2018) focuses on these applications of neural networks while

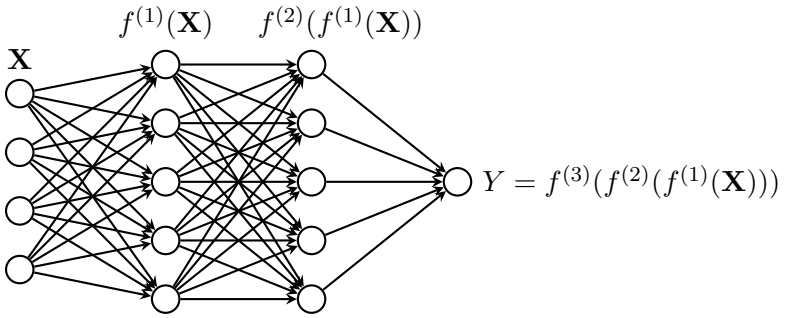

Figure 1.3: Example of a feedforward neural network with 3 layers.

Deng and Liu (2018) recompile state of the art of NLP-centric deep learning research. Convolutional neural networks are widely used in computer vision with great success in classification of natural objects. Tasks such as pedestrian detection, car detection, traffic sign recognition and traffic light (see Aghdam and Heravi (2017) for a deeper introduction on convolutional neural networks and their applications). Aggarwal et al. (2018) and Goodfellow et al. (2016) offer an excellent overview for neural networks while Skansi (2018) gives a detail review of the history of neural networks.

1.4 Dimensionality Reduction

In unsupervised learning there is no label associated to each observation in the data set. Instead, in these problems we want to learn about the structure of the features. Examples of unsupervised learning are density estimation, where we want to estimate the entire probability distribution of the features; clustering, where we are interested in dividing the data set into clusters of similar observations; or dimensionality reduction, where we want to find a space of smaller dimension than the input space to represent the data. We are going to focus on the latter.

Dimensionality reduction is the process of reducing the number of features that describe some data which can be useful in many situations that require low dimensional data (data visualization, data storage, heavy computation, etc.). This reduction is done either by selection (only some existing features are conserved) or by extraction (a reduced number of new features are created based on the old features). Following the notation and definitions introduced by Rocca (2019), let us call encoder the process that produce the new representation from the "old features" (by selection or by extraction) and decoder the reverse process. Dimensionality reduction can then be interpreted as data compression where the encoder compress the data (from the initial space to the encoded space, also called latent space) whereas the decoder decompress them. The main purpose of a dimensionality reduction method is to find the best encoder/decoder pair among a given family. In other words, for a given set of possible encoders and decoders, we are looking for the pair that keeps the maximum of information when encoding and, so, has the minimum of reconstruction error when decoding. If we denote respectively as $\mathcal{E}$ and $\mathcal{D}$ the families of encoders and decoders we are considering, then the learning process is described simply as minimizing a loss function, that is

$$(e^\star, d^\star) \in \underset{(e,d)\in\mathcal{E}\times\mathcal{D}}{\arg\min}\ L(\mathbf{X}, d(e(\mathbf{X})))$$

1.4.1 Principal Component Analysis (PCA)

The idea of PCA is to build k new independent features that are linear combinations of the p old features ($k \leq p$) and so that the projections of the data on the subspace defined by these new features are as close as possible to the initial data (in terms of euclidean distance). In other words, PCA is looking for the best linear subspace of the initial space (described by an orthogonal basis of new features) such that the error of approximating the data by their projections on this subspace is as small as possible.

Translated in our global framework, we are looking for an encoder in the family $\mathcal{E}$ of the $p \times k$ matrices (linear transformation) whose rows are orthonormal (features independence) and for the associated decoder among the family $\mathcal{D}$ of $p \times k$ matrices. It can be shown that the unitary eigenvectors corresponding to the k greatest eigenvalues (in norm) of the covariance features matrix are orthogonal (or can be chosen to be so) and define the best subspace of dimension k to project data on with minimal error of approximation. If we denote by V_k this matrix, then the PCA loss function is

$$V_k = \underset{V \in M_{p \times k}}{\arg\min} \|\mathbf{X} - VV^T\mathbf{X}\|_2^2$$

we can recognize $\mathbf{Z} = V^T\mathbf{X}$ as the encoder portion and $\mathbf{Y} = V\mathbf{Z}$ as the decoder portion.

1.4.2 Autoencoders

An autoencoder is a neural network that attempts to reconstruct its input to its output. Internally, there is a hidden layer $\mathbf{Z}$, called the code, that describes the input. Hence, the network may be viewed of consisting of two parts: an encoder function $\mathbf{Z} = e(\mathbf{X})$ and a decoder function $\mathbf{Y} = d(\mathbf{Z})$. Reconstructing the data might seem trivial by simply copying the data forward from one layer to another or simply allowing the network to set $d(e(\mathbf{X})) = \mathbf{X}$, in both cases the autoencoder is not especially useful. Instead, autoencoders are designed to avoid copying perfectly its input to its output. To avoid just copying the data forward from one layer to another the code layer $\mathbf{Z}$ is constrained to have a smaller dimension than $\mathbf{X}$, and thus avoiding (at least partially) a perfect reconstruction of the input. An autoencoder whose code dimension is less than the input dimension is called undercomplete.

Traditionally, autoencoders have been used for dimensionality reduction. Thus, even when training a network to learn a *lossy* reconstruction of the data may sound useless, we hope that the code $\mathbf{Z}$ will learn useful features of the data. Learning an undercomplete representation forces the autoencoder to capture the important characteristics that describe the training data.

It is common (but not necessary) for an $M-$layer autoencoder to have a symmetric architecture between the input and the output, where the number of units in the $k-$th layer is the same that in the $(M-k+1)$-th layer. The value of M is often odd, and the $(M+1)/2-$th layer is often the most constricted one. Therefore, the minimum number of layers in an autoencoder is three, corresponding to the input layer, the code layer and the output layer. However, the real power of autoencoders is achieved when deeper variants are used. Figure 1.4 shows a diagram of an autoencoder with three layers.

Note that the output $\mathbf{Y}$ of the autoencoder does not necessarily belongs to the same space than $\mathbf{X}$. Thus a final function $\widehat{\mathbf{X}} = f(\mathbf{Y})$ might be needed to transform the output of the decoder into the final reconstruction of $\mathbf{X}$. This function is completely determined and is independent of the autoencoder. For example, it could be that all the entries of $\mathbf{X}$ are zero or one, that is $\mathbf{X} \in \{0,1\}^p$ and we could be at the end of the neural networks with an output $\mathbf{Y}$ whose values are taken in $[0,1]^p$ (for example when

a sigmoid function is used for each neuron of the output layer), then we can choose $f(\mathbf{Y}^{(h)}) = \mathbb{1}_{\mathbf{Y}^{(h)} \geq 0.5}$ in order to recover binary values.

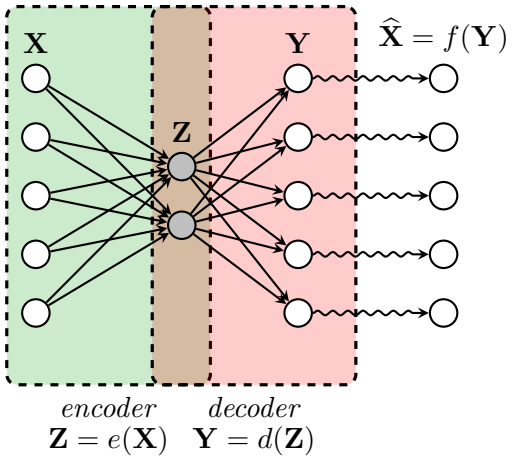

Figure 1.4: Diagram of an autoencoder with three layers. The latent space $\mathbf{Z}$ forms a lower dimensional representation of the input $\mathbf{X}$. The autoencoder then decodes $\mathbf{Z}$ attempting to reconstruct $\mathbf{X}$ from this representation.

When the autoencoder has a single hidden layer (the code), the encoder is linear and the loss function L is the mean squared error, an undercomplete autoencoder would learn the same subspace as PCA (Aggarwal et al., 2018). Autoencoders with nonlinear encoder function can thus learn a more powerful nonlinear dimensionality reduction.

Although several methods for nonlinear dimensionality reduction are known, autoencoders allow a lot of flexibility by varying the number of layers and their characteristics such as the number of units and type of activation functions used in intermediate stages. Unfortunately, if the encoder and decoder have too much flexibility, the autoencoder can learn to perform the copying task without extracting useful information of the data (Aggarwal et al., 2018).

Autoencoders (LeCun, 1987; Bourlard and Kamp, 1988; Hinton and Zemel, 1993) or regularized versions as sparse autoencoders have been used for dimensionality reduction, feature learning and compression while denoising autoencoders have been used to reconstruct good examples from corrupted data. Variational Autoencoders (VAEs) (Kingma and Welling, 2013) keep an association with classsical autoencoders due to their architectural affinity, however as Generative Adversarial Networks (GANs) (Goodfellow et al., 2014), these are generative models which attempt to learn a distribution from which they can sample new objects that are similar to those in the training data set. Kingma and Welling (2019), Doersch (2016), and Rocca (2019) give an introducction to VAEs and Creswell et al. (2018) offer an overview for GANs.

1.5 Mechanisms for Missing Data

Up to now we have assumed that the entries of the data matrix are observed, however in practice there could be "blank" spaces in the data set. For example, respondents in a household survey may refuse to report income. In an industrial experiment some results are missing because of mechanical breakdowns unrelated to the experimental process. In an opinion survey some individuals may be unable to express a preference for one candidate over another, etc. Standard statistical methods typically exclude units that

have missing value for any of the variables involved in an analysis. This strategy is generally inappropriate, since the investigator is usually interested in making inferences about the entire target population, rather than the portion of the target population that would provide responses on all variables in the analysis.

It is also relevant to consider the mechanisms that lead to missing data, and in particular the question of whether the fact that variables are missing is related to the underlying values of the variables in the data set. The concept of data-missing mechanisms (introduced by Rubin (1976)) establishes the relationship between missingness and data. It is common to define the data-missing mechanisms through the data matrix. However, to have a useful definition from a theoretical point of view, we formally define them using the random variables $\mathbf{X}$ and Y. First, let us define a new variable, called the indicator of missing value,

$$\mathbf{M}^{(h)} = \begin{cases} 1 & \text{if } \mathbf{X}^{(h)} \text{ is missing} \\ 0 & \text{otherwise} \end{cases}, \quad 1 \leq h \leq p$$

Since we are interested in the relation between the target Y and the predictor variables $\mathbf{X}$, there is no point to include observations were the target is missing since they would be uninformative. Hence, we are assuming throughout this work that the response Y has no missing values which makes unnecessary to define an indicator of missing variable for Y. Then, the mechanisms are fully characterized by the information of the conditional distribution of $\mathbf{M}^{(h)}$ given $(\mathbf{X}, Y)$. There are three possibly data-missing mechanisms.

Missing Completely at Random (MCAR) We say that the variable $\mathbf{X}^{(h)}$ is MCAR if $\mathbf{M}^{(h)} \perp (\mathbf{X}, Y)$. In other words, under the MCAR assumption, a coordinate $\mathbf{X}^{(h)}$ has some probability to be missing in the sample and this probability does not depend on the value of $\mathbf{X}$ nor the response variable Y.

Missing at Random (MAR) Let us define the set $h_o = \{h : \mathbb{P}\left[\mathbf{M}^{(h)} = 0\right] = 1\}$, thus $\mathbf{X}^{(h_o)}$ is the vector conformed by those variables that are always observed. The variable $\mathbf{X}^{(h)}$ is MAR if

$$\mathbb{P}\left[\mathbf{M}^{(h)}(\mathbf{X}) = 1 | (\mathbf{X}, Y)\right] = \mathbb{P}\left[\mathbf{M}^{(h)}(\mathbf{X}) = 1 | (\mathbf{X}^{(h_o)}, Y)\right]$$

That is, the probability that $\mathbf{X}^{(h)}$ is missing just depends on observed data.

Not Missing at Random (NMAR) If the probability of missingness depends on unobserved values we say that $\mathbf{X}^{(h)}$ is NMAR.

For example, suppose that $p = 2$, $\mathbf{X}^{(1)}$ is the age, and $\mathbf{X}^{(2)}$ is the income. If the probability for income being missing is the same for all individuals, regardless of their age or income, then the data is MCAR. If the probability for income being missing varies according to the age of the respondent but does not vary according to the income of respondents with the same age, then the data is MAR. If the probability for income being recorded varies according to income itself, then the data is NMAR. This latter case is the hardest to deal with analytically, which is unfortunate, since it may be the most likely case in many applications. As commented by Little and Rubin (2002) in the controlled missing-data environment of double sampling, the missing data is MAR. Double sampling refers to survey designs in which initially a sample of units is selected for obtaining auxiliary information only, and then a subsample is selected in which the variable of interest is observed in addition to the auxiliary information.

The purpose of double sampling is to obtain better estimators by using the relation between auxiliary variables and the target variable. For interested readers in sampling we recommend the monograph of Thompson (2012). The example of missingness in an industrial experiment due to a mechanical breakdown is a case a MAR mechanism. On the other hand a scenario of censored data is a typical case of an NMAR mechanism.

1.6 Previous Approaches to Handle Missing Data Using Random Forests

1.6.1 Previous Approaches Implementing Imputation of Missing Values

Many methods proposed in the literature to handle missing data using random forests operate through imputation in a recursive way. First, they use the original training data set $\mathcal{D}_n$ to fill the blank spaces in a roughly way. For example, with the median of the observed values in the variable. We denote this new data set as $\mathcal{D}_{n,1}$.

The imputed data set $\mathcal{D}_{n,1}$ is used to build a random forest. Then, some structures of the forest are exploited, like the so-called proximity matrix, improving the imputation and resulting in a new data set $\mathcal{D}_{n,2}$. The procedure follows iteratively until some stopping rule is applied, for example when there is little change between the imputed values or when a fix number of iterations is achieved. More formally, let us define

$$\mathbf{X}_{i,\ell}^{(h)} = \begin{cases} \mathbf{X}_i^{(h)} & \text{if } \mathbf{M}_i^{(h)} = 0 \\ \widehat{\mathbf{X}}_{i,\ell}^{(h)} & \text{if } \mathbf{M}_i^{(h)} = 1 \end{cases}$$

where $\widehat{\mathbf{X}}_{i,\ell}^{(h)}$ is the imputation of $\mathbf{X}_i^{(h)}$ at time $\ell \geq 1$, and let $\mathbf{X}_{i,\ell} = \left(\mathbf{X}_{i,\ell}^{(1)}, \ldots, \mathbf{X}_{i,\ell}^{(p)}\right)$.

In order to properly introduce previous methods that handle missing data through imputation, we need to define the connectivity between two points in a tree and the proximity matrix of the forest. Let $K_{\Theta,n}(\mathbf{X}, \mathbf{X}') = \mathbb{1}_{\mathbf{X} \overset{\Theta}{\leftrightarrow} \mathbf{X}'}$ be the indicator that $\mathbf{X}$ is in the same final cell that $\mathbf{X}'$ in the tree designed with $\mathcal{D}_n$ and the parameter Θ. If $K_{\Theta,n}(\mathbf{X}, \mathbf{X}') = 1$ we say that $\mathbf{X}$ and $\mathbf{X}'$ are connected in the tree $m_n(\cdot; \Theta)$. The proximity of two points is the average of times in which they are connected in the forest, it measures the similarity of the observations in the eyes of the random forest. Formally, let us define the proximity between $\mathbf{X}$ and $\mathbf{X}'$ in the finite forest, $m_{M,n}(\cdot; \Theta_1, \ldots, \Theta_M)$, as

$$K_{M,n}(\mathbf{X}, \mathbf{X}') = \frac{1}{M} \sum_{k=1}^{M} K_{\Theta_k,n}(\mathbf{X}, \mathbf{X}')$$

To simplify the notation, let $K_{M,\ell}(i,j)$ be the proximity between $\mathbf{X}_i$ and $\mathbf{X}_j$ at time ℓ. That is, $K_{M,\ell}(i,j)$ is the proximity between $\mathbf{X}_i$ and $\mathbf{X}_j$ in the random forest constructed with the data set $\mathcal{D}_{n,\ell}$. We also define $\mathbf{i}_{miss}^{(h)} \subseteq \{1, \ldots, n\}$ as the indexes where $\mathbf{X}^{(h)}$ is missing, and $\mathbf{i}_{obs}^{(h)} = \{1, \ldots, n\} \setminus \mathbf{i}_{miss}^{(h)}$ as the indexes were $\mathbf{X}^{(h)}$ is observed.

Breiman's Approach This method to handle missing values was proposed by Breiman (2003). If $\mathbf{X}^{(h)}$ is a continuous variable, $\widehat{\mathbf{X}}_{j,\ell+1}^{(h)}$ is the weighted mean of the observed values in $\mathbf{X}^{(h)}$, where the weights are defined by the proximity matrix of the

previous random forest, that is

$$\widehat{\mathbf{X}}_{j,\ell+1}^{(h)} = \frac{\sum_{i\in\mathbf{i}_{obs}^{(h)}} K_{M,\ell}(i,j)\mathbf{X}_i^{(h)}}{\sum_{i\in\mathbf{i}_{obs}^{(h)}} K_{M,\ell}(i,j)}, \qquad \begin{array}{l}\ell \geq 1\\ j \in \mathbf{i}_{miss}^{(h)}\end{array}$$

On the other hand, if $\mathbf{X}^{(h)}$ is a categorical variable, $\widehat{\mathbf{X}}_{j,\ell+1}^{(h)}$ is given by

$$\widehat{\mathbf{X}}_{j,\ell+1}^{(h)} = \underset{\mathbf{x}\in\mathcal{X}^{(h)}}{\arg\max} \sum_{i\in\mathbf{i}_{obs}^{(h)}} K_{M,\ell}(i,j)\mathbb{1}_{\mathbf{X}_i^{(h)}=\mathbf{x}}, \qquad \begin{array}{l}\ell \geq 1\\ j \in \mathbf{i}_{miss}^{(h)}\end{array}$$

That is, $\widehat{\mathbf{X}}_{j,\ell+1}^{(h)}$ is the class that maximizes the sum of the proximity considering the observed values in the class.

Ishioka's Approach This method, proposed by Ishioka (2013), is an improvement to outliers of the previous Breiman's approach. If $\mathbf{X}^{(h)}$ is a continuous variable, $\widehat{\mathbf{X}}_{j,\ell+1}^{(h)}$ is the weighted mean of the k nearest neighbors, according to the proximity matrix, over all the values, both imputed and observed. The k closest values are chosen to make more robust the method and avoid values which are outliers.

$$\widehat{\mathbf{X}}_{j,\ell+1}^{(h)} = \frac{\sum_{\substack{i\in \text{neigh}_k \\ i\neq j}} K_{M,\ell}(i,j)\widehat{\mathbf{X}}_{i,\ell}^{(h)}}{\sum_{\substack{i\in \text{neigh}_k \\ i\neq j}} K_{M,\ell}(i,j)}, \qquad \begin{array}{l}\ell \geq 1\\ j \in \mathbf{i}_{miss}^{(h)}\end{array}$$

For categorical variables, it is not necessary to see only the k closest values because the outliers of $\mathbf{X}$ will have few attention. Meanwhile the proximity with missing values should have more attention, specially when the missing rate is high. Hence, if $\mathbf{X}^{(h)}$ is a categorical variable, $\widehat{\mathbf{X}}_{j,\ell+1}^{(h)}$ is given by

$$\widehat{\mathbf{X}}_{j,\ell+1}^{(h)} = \underset{\mathbf{x}\in\mathcal{X}^{(h)}}{\arg\max} \sum_{i\neq j} K_{M,\ell}(i,j)\mathbb{1}_{\widehat{\mathbf{X}}_{i,\ell}^{(h)}=\mathbf{x}}, \qquad \begin{array}{l}\ell \geq 1\\ j \in \mathbf{i}_{miss}^{(h)}\end{array}$$

MissForest This algorithm, proposed by Stekhoven and Bühlmann (2011), handle the missing data as a supervised learning problem itself, where the target variable is the input variable with missing values. The MissForest consists in iteratively building a random forest from the observed data and the previous imputations to predict the missing values of the input variables.

1.6.2 Previous Approaches Without Implementing Imputation of Missing Values

A simple idea to handle missing values is to use all the observations and dropping out those that have the split variable missing. An obvious problem of this alternative is that we can finish with a relatively small amount of data very quickly. However, if we have already constructed the tree and we want to predict a new observation that has missing values in some features, we can do it with this approach. For prediction, it consists in dropping the case down the tree as far as it will go, until we cannot longer assign the observation to one of the child nodes. In the same spirit, for binary trees we can change the structure of the trees and consider ternary splits instead of binary, where the third child contains all observations where the feature is missing

A straightforward option to handle missing variables in categorical predictors is creating a new "missing" category. Quinlan (1986) presents an example in which missing values are introduced into a feature creating an extra "missing" category, then it is shown that just by creating this new category the apparent gain in information increases. This example shows that this approach can create the anomalous situation in which a feature with missing values is preferred over features with all the observations, a result entirely opposed to common sense. Another simple choice is to send all incomplete observations to a side chosen by minimizing the error.

Quinlan (1993) propose to assign to each observation in a cell A a weight for each child node representing the probability that the observation belongs to that node. Then dividing the observations into fractional objects, in this case the cardinality of each node should be reinterpreted as the sum of the fractional weights of the observations in the node. Furthermore, Quinlan (1993) suggests how to use these weights for predicting a new observation for a classification task. Suppose that the observation has weight w in cell A which has child nodes $A_1, \ldots, A_T$ and that $c_1, \ldots, c_T$ are the classes associated to each one of the child nodes, that is

$$c_t \in \arg\max_{c_j \in \mathcal{Y}} \frac{1}{N(A)} \sum_{i=1}^{n} \mathbb{1}_{Y_i = c_j, \mathbf{X}_i \in A_t}.$$

Then the weight for the cell A_t is given by $w_t = w \times p_{c_t}$, the observation explores all the branches and it is classified in the class with the highest probability.

A popular alternative consists in the so-called surrogate splits (Breiman et al., 1984). As described by Friedman, Hastie, and Tibshirani (2009), when considering a variable to split, we use only the observations for which that variables is not missing. Once we have selected the best cut $(\widehat{h}, \widehat{z})$ we form a list of surrogate variables and positions of cuts $(h_1, z_1), \ldots, (h_s, z_s)$, $\widehat{h} \neq h_1 \neq \cdots \neq h_s$, where the first surrogate split (h_1, z_1) corresponds to the best cut that mimics the split $(\widehat{h}, \widehat{z})$. The second surrogate split (h_2, z_2) corresponds to the second best and so on. The surrogate splits are found by applying the partition algorithm to predict the two categories $\{\mathbf{X}^{(\widehat{h})} < \widehat{z}\}$ and $\{\mathbf{X}^{(\widehat{h})} \geq \widehat{z}\}$ using the rest of the variables. An important advantage of surrogate splits is that they can be used to send observations down the tree either during training or during prediction. The default behaviour of the `rpart` library in `R` is to find surrogate splits during tree construction, and use them if missing values are found during prediction. This can be changed by the option `usesurrogate = 0` to stop cases as soon as a missing attribute is encountered. A further choice is what do to if all surrogates are missing: option `usesurrogate = 1` stops whereas `usesurrogate = 2` (the default) sends the case in the majority direction, (see Venables and Ripley, 2002; Therneau, Atkinson, et al., 1997). Despite its widespread acceptance and application there is only little published knowledge about its performance. Feelders (1999) and Farhangfar, Kurgan, and Dy (2008) compare the differences between the use of surrogate splits and multiple imputation (Rubin, 1996; Rubin, 2004). However these works lack generalizability as modeling is restricted to classification tasks, categorical data and special simulation schemes. More recently Hapfelmeier, Hothorn, and Ulm (2012) concluded that there was not a clear improvement by using fully conditional specification (FCS) also known as imputation with conditional equations (ICE) (Van Buuren et al., 2006). Rieger, Hothorn, and Strobl, 2010 compare k-nearest neighbor (kNN) imputation and surrogate splits on different data-missing mechanisms and for classification and regression problems.

The Missing Incorporated in Attributes (MIA) approach was introduced by Twala, Jones, and Hand (2008). This method uses the missing values to compute the split

and hence, it does not require a different algorithm to propagate the observations with missing values down the tree. This algorithm keeps the observations with missing values together and the candidate splits assign the values according to one of the following rules:

- $\{\mathbf{X}^{(h)} < z \text{ and } \mathbf{M}^{(h)} = 1\}$ versus $\{\mathbf{X}^{(h)} \geq z\}$
- $\{\mathbf{X}^{(h)} < z\}$ versus $\{\mathbf{X}^{(h)} \geq z \text{ and } \mathbf{M}^{(h)} = 1\}$
- $\{\mathbf{M}^{(h)} = 0\}$ versus $\{\mathbf{M}^{(h)} = 1\}$

Josse et al. (2019) perform a simulation study to compare different methods to handle missing data using trees and study the consistency of two approaches to estimate the prediction function with missing values in a framework of multiple imputation, the use of a universally consistent algorithm and for a Missing At Random (MAR) mechanism. Furthermore, from simulation studies they concluded that MIA is a good alternative to handle missing values. Many of these algorithms have been implemented in `R` in the package `partykit` (Hothorn and Zeileis, 2015)

1.7 A Random Forest Algorithm with Partial Imputations for Missing Entries

Let us emphasize, through a toy example, the issues that one faces with the original CART criterion in a context of missing values. Consider the space $\mathcal{X} = [0,1]^2$ and two observations, $\mathbf{X}_1$ and $\mathbf{X}_2$, that belong to a cell $A \subset \mathcal{X}$. We assume that a direction and location for a cut of A have been chosen and we denote by A_L and A_R the two resulting cells, Table 1.1 shows the toy example data and Figure 1.5 an illustration. $\mathbf{X}_1$ is represented as a dashed line since $\mathbf{X}_1^{(1)}$ is missing[1] over the interval $[0,1]$.

	$\mathbf{X}^{(1)}$	$\mathbf{X}^{(2)}$
$\mathbf{X}_1$	`NA`	0.5
$\mathbf{X}_2$	0.75	0.25
A	[0.3,0.9]	[0.2,0.7]
A_L	[0.3,0.6]	[0.2,0.7]
A_R	[0.6,0.9]	[0.2,0.7]

Table 1.1: Toy example data, we consider a cell A that has been split into A_L and A_R. The variable $\mathbf{X}^{(1)}$ is missing in the first observation (denoted as `NA`).

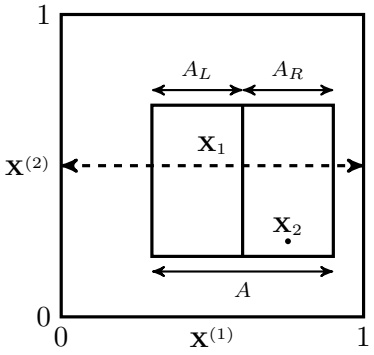

Figure 1.5: Illustration of the example data, $\mathbf{X}_1$ is represented as a dashed line since $\mathbf{X}^{(1)}$ is missing.

It is clear that $\mathbf{X}_2 \in A_R$, however it is not possible to decide, without any further operation, if $\mathbf{X}_1 \in A_R$ or $\mathbf{X}_1 \in A_L$, or even if $\mathbf{X}_1 \in A$. The CART criterion is then intractable since the quantities $N(A)$, $N(A_L)$, $N(A_R)$, $\bar{Y}_A$, $\bar{Y}_{A_L}$, $\bar{Y}_{A_R}$, $\mathbb{1}_{\mathbf{X}_i \in A}$, $\mathbb{1}_{\mathbf{X}_i^{(h)} < z}$ and $\mathbb{1}_{\mathbf{X}_i^{(h)} \geq z}$ (highlighted in Equation (1.2)) cannot be computed.

[1] In our illustrations we represent as dashed lines observations with missing values.

$$
\begin{aligned}
L_{n,A}(h,z) = & \frac{1}{N(A)} \sum_{i=1}^{n} \left(Y_i - \bar{Y}_A\right)^2 \mathbb{1}_{\mathbf{X}_i \in A} \\
& - \frac{1}{N(A)} \sum_{i=1}^{n} \left(Y_i - \bar{Y}_{A_L}\right)^2 \mathbb{1}_{\mathbf{X}_i \in A, \mathbf{X}_i^{(h)} < z} \\
& - \frac{1}{N(A)} \sum_{i=1}^{n} \left(Y_i - \bar{Y}_{A_R}\right)^2 \mathbb{1}_{\mathbf{X}_i \in A, \mathbf{X}_i^{(h)} \geq z}
\end{aligned} \tag{1.2}
$$

Proposed Approach: The approach proposed in this thesis keeps the form of the CART criterion and makes use of adapted imputations for the intractable parts that allow us to compute the CART criterion. Unlike most of the so-called imputation techniques, the imputation step is not performed independently of the evaluation of the CART criterion but is integrated to its later optimization. As a cut is chosen as a maximiser of the CART criterion in the normal set up, now a couple (cut, imputation) is chosen at each split in the creation of the random tree. The idea is that, for a cut, the observations with missing values are assigned to the child node that maximizes our CART criterion. Then, we move on to the next cell and proceed in the same way until a stopping rule is achieved. At the end the missing observations would end into a final node of the tree, which can give an "imputation" of the missing values as a region of the input space, which in turn would be translated into a "cloud" of possible regions for the missing values when the random forest is consider. With some extra effort we believe that is possible to translate these regions into confidence regions for the missing values.

This algorithm looks similar to other approaches that perform imputation. As an initialization, we "impute" the missing values with all the possible values of the variable, then at each split the missing values locations are updated belonging to one of the child nodes. Despite this similarity, our algorithm does not impute the missing places with a punctual value, instead each tree "imputes" the missing places with an interval. Note also, that the algorithm proposed in this work has the advantage of not having to calculate extra structures like the proximity matrix. Furthermore, we do not need to change the imputation of the variables, instead we refine them during the descend process. To avoid confusion in the sequel, we make the subtle difference between the imputation of the missing values at the start (referred to as *in*) of an iteration and the imputation at the end (referred to *out*) of the iteration. The mathematical formalism is as follows.

- $\widehat{\mathbf{X}}_{i,in} = \left(\widehat{\mathbf{X}}_{i,in}^{(1)}, \ldots, \widehat{\mathbf{X}}_{i,in}^{(p)}\right)$ is the current imputation of $\mathbf{X}_i$.
- $\widehat{N}(A)$ is the number of points assigned to the cell A.
- $\widehat{Y}_A$ is the empirical mean of the response variable Y_i such that $\widehat{\mathbf{X}}_{i,in}$ belongs to the cell A.
- $\widehat{N}_{miss}^{(h)}(A) = \sum_{i=1}^{n} \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, \mathbf{M}_i^{(h)}=1}$ is the number of observations assigned to the cell A whose variable $\mathbf{X}^{(h)}$ is missing. We denote by $\widehat{N}_{miss}(A)$ the number of observations assigned to A such that at least one coordinate is missing and by $\widehat{N}_{obs}(A)$ the number of observations with no missing values inside the cell A.

- $\mathbf{i}_{A,miss}^{(h)} = \{j_1, \ldots, j_{\widehat{N}_{miss}^{(h)}(A)}\}$ is the set of indexes of the observations assigned to the cell A whose variable $\mathbf{X}^{(h)}$ is missing.
- For each direction h, let $\mathcal{W}_A^{(h)} = \{0,1\}^{\widehat{N}_{miss}^{(h)}(A)}$ be the collection of binary vectors w with the convention that $w_k = 1$ means that the observation $(\mathbf{X}_{j_k}, Y_{j_k})$ is assigned to the left child node and $w_k = 0$ means that the observation $(\mathbf{X}_{j_k}, Y_{j_k})$ is assigned to the right child node, for all $j_k \in \mathbf{i}_{A,miss}^{(h)}$.

Finally, the variables $\mathbf{X}_{j_k}^{(h)}$ with $j_k \in \mathbf{i}_{A,miss}^{(h)}$ are updated by

$$\widehat{\mathbf{X}}_{j_k,out}^{(h)} = \begin{cases} \left[a^{(h)}, z\right] & \text{if } w_k = 1 \\ \left[z, b^{(h)}\right] & \text{if } w_k = 0 \end{cases}$$

while $\widehat{\mathbf{X}}_{i,out}^{(h)} = \widehat{\mathbf{X}}_{i,in}^{(h)}$ for all $i \in \{1, \ldots, n\} \backslash \mathbf{i}_{A,miss}^{(h)}$ and every other coordinate is kept unchanged in the process, so that, for all $h' \neq h$, $\forall i$, $\widehat{\mathbf{X}}_{i,out}^{(h')} = \widehat{\mathbf{X}}_{i,in}^{(h')}$. To keep the notation fairly simple, we omit the dependence on w in the notation of $\widehat{\mathbf{X}}_{out}$ even when the two notions are deeply linked by definition. See Figure 1.6 for an illustration with $p = 2$.

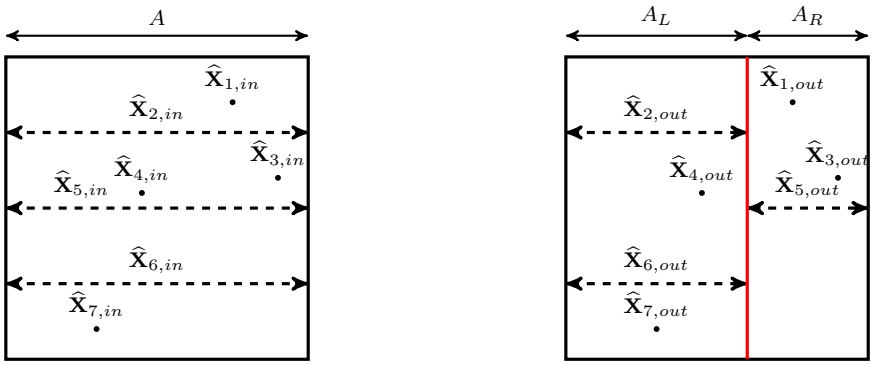

Figure 1.6: We perform a cut and assignation of points where the variable is missing.

The empirical CART criterion, on a cell A, in the context of missing values is defined as

Definition 3 Empirical CART Criterion with Missing Values

$$\begin{aligned} L_n(A,d,w) = & \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_A\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A} \\ & - \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_L}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, a^{(h)} \leq \widehat{\mathbf{X}}_{i,out}^{(h)} < z} \\ & - \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_R}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, z \leq \widehat{\mathbf{X}}_{i,out}^{(h)} \leq b^{(h)}} \end{aligned}$$

where $\widehat{Y}_{A_L}$ (respectively $\widehat{Y}_{A_R}$) is the empirical mean of the Y_i such that $\widehat{\mathbf{X}}_{i,out}$ belongs to the cell A_L (respectively A_R).

For a cell A and an input imputation vector $\widehat{\mathbf{X}}_{in}$, the algorithm chooses a cut and assignation $(\widehat{d}, \widehat{w})$ by maximizing $L_n(A, d, w)$ over $\mathcal{C}_A \times \mathcal{W}_A^{(h)}$,

Definition 4 Best Empirical Cut and Assignation

$$(\widehat{d}, \widehat{w}) \in \underset{\substack{d \in \mathcal{C}_A \\ w \in \mathcal{W}_A^{(h)}}}{\arg\max}\, L_n(A, d, w)$$

Finally, the "imputed" intervals are updated, which we symbolize by $\widehat{\mathbf{X}}_{i,in}^{(\widehat{h})} \leftarrow \widehat{\mathbf{X}}_{i,out}^{(\widehat{h})}$.

Note that, if we know the mechanism of missingness, the possible values for w might change. For example, assume that there are missing values just in $\mathbf{X}^{(1)}$, and its value is missing when it is bigger than τ, that is, we have right censoring. Also consider that we know the value of τ, so the missing mechanism is known. if we perform a cut at τ in $\mathbf{X}^{(1)}$, then the only admissible assignation for $\widehat{\mathbf{X}}_{i,out}$, where $i \in \mathbf{i}_{A,miss}^{(h)}$, is to the right child node.

From a theoretical point of view, we do not longer have observations, hence we introduce the notions of the input assigned random variable $\widehat{\mathbf{X}}_{in}$ and output assigned random variable $\widehat{\mathbf{X}}_{out}$. The random variable $\widehat{\mathbf{X}}_{in}$ corresponds to a prior assignation whereas $\widehat{\mathbf{X}}_{out}$ corresponds to the distribution of the assignation after the theoretical cut is performed. Furthermore, the binary assignations w are translated into probabilities.

Formally, let $\mathcal{W}$ be the collection of functions from $\mathcal{Y}$ to $[0, 1]$, and $\widehat{\mathbf{X}}_{in} = (\widehat{\mathbf{X}}_{in}^{(1)}, \ldots, \widehat{\mathbf{X}}_{in}^{(p)})$ be the input distribution for the imputation, then $\widehat{\mathbf{X}}_{out}^{(h)} | \widehat{\mathbf{X}}_{in}$ is defined (in distribution) as

Definition 5 Output Assigned Random Variable

$$\widehat{\mathbf{X}}_{out}^{(h)} | \widehat{\mathbf{X}}_{in} \in A = \begin{cases} \mathbf{X}^{(h)} | \mathbf{X} \in A & \text{if } \mathbf{M}^{(h)} = 0 \\ \mathbf{B}^{(h)} & \text{if } \mathbf{M}^{(h)} = 1 \end{cases}$$

where

$$\mathbf{B}^{(h)} = \begin{cases} \left(a^{(h)}, z\right) & \text{if } \mathrm{Ber}(w(Y)) = 1 \\ \left(z, b^{(h)}\right) & \text{if } \mathrm{Ber}(w(Y)) = 0 \end{cases}, \quad w \in \mathcal{W}$$

The imputation variable $w(Y)$ is the (random) probability that $\widehat{\mathbf{X}}_{out}^{(h)} < z$ conditionally to $\mathbf{M}^{(h)} = 1$ and Y. Note that $\widehat{\mathbf{X}}_{in}$ always belongs to a cell A, so the above definition of $\widehat{\mathbf{X}}_{out}^{(h)} | \widehat{\mathbf{X}}_{in} \in A$ is well defined. We define the theoretical CART over a cut $d = (h, z)$ and a function $w \in \mathcal{W}$ as

Definition 6 Theoretical CART Criterion with Missing Values

$$\begin{aligned} L^\star(A, d, w) = &\mathbb{V}[Y | \widehat{\mathbf{X}}_{in} \in A] - \mathbb{V}[Y | \widehat{\mathbf{X}}_{out}^{(h)} < z, \widehat{\mathbf{X}}_{in} \in A] \mathbb{P}[\widehat{\mathbf{X}}_{out}^{(h)} < z | \widehat{\mathbf{X}}_{in} \in A] \\ &- \mathbb{V}[Y | \widehat{\mathbf{X}}_{out}^{(h)} \geq z, \widehat{\mathbf{X}}_{in} \in A] \mathbb{P}[\widehat{\mathbf{X}}_{out}^{(h)} \geq z | \widehat{\mathbf{X}}_{in} \in A] \end{aligned}$$

and the best cut and assignation $(d^\star, w^\star)$ is selected by maximizing $L^\star(A, d, w)$ over $\mathcal{C}_A \times \mathcal{W}$, that is

Definition 7 Best Theoretical Cut and Assignation

$$(d^\star, w^\star) \in \underset{\substack{d \in \mathcal{C}_A \\ w \in \mathcal{W}}}{\arg\max}\, L^\star\left(A, d, w\right)$$

2

Simulation Study of a Random Forest Algorithm with Interval Imputation of Missing Entries

This chapter presents a simulation study for our proposed method to handle missing values in a regression learning task. The previous work of Rieger, Hothorn, and Strobl (2010) is taken as a basis for this study due to the extensive number of mechanisms of missingness considered. The study here presented compare our proposal with other 7 methods that handle missing data using random forests. Three of them are taken as simple baselines, corresponding to median imputation, listwise deletion (observations with at least one missing value are deleted from the data set), and the elimination of variables with missing values. Three methods correspond to more elaborate imputation algorithms corresponding to the Breiman's method presented in Breiman (2003), the improvement suggested by Ishioka (2013) (here presented simply as Ishioka's method) and missForest (Stekhoven and Bühlmann, 2011). The last method considered in this study correspond to Missing Incorporated in Attributes (MIA) (Twala, Jones, and Hand, 2008) which uses the observations with missing values directly in the construction of the recursive trees. Two of these approaches, corresponding to missForest and MIA could be considered as state-of-the-art algorithms to handle missing values through random forests. As Rieger, Hothorn, and Strobl (2010), we compare the methods' performance based on their mean squared error (MSE), but also in terms of their bias. Furthermore, we vary the number of observations with some missing value between 5% of the data set to 95%, comparing the mentioned methods under all these circumstances.

At the end of this chapter we break the CART criterion used in the construction of our random forests, giving meaningful interpretation to its components and studying their behavior when the missing values are incorporated accordingly to an MCAR procedure.

All the simulations were done in `R`. The package used for the missForest method was the `missForest` package. Since no package seems to handle missing values using the CART criterion and the MIA approach, this procedure as well as our proposal were completely programmed from scratch. For the rest of the methods we have used the `randomForests` package.

2.1 Bibliographic Discussion

Handle missing values through different algorithms has taken attention for the last decades with several simulation studies comparing the performance of distinct methods. Feelders (1999) is one of the first to present a result on missing values using recursive

trees. This work compares the error rate between surrogate splits in a single decision tree (implemented in the `R` package `rpart` (Therneau, Atkinson, et al., 1997)) and the imputation procedure presented by Schafer (1997) and Schafer and Olsen (1998). It considers two data sets in the study, given by the so-called *Waveform* data set (presented by Breiman et al. (1984)) and the *Prima Indian Datasets* (available at the UCI machine learning repository (Dua and Graff, 2017)). The percentage of missing values varies between 10% and 45% for the first data set, where the missing values are introduced accordingly to an MCAR mechanism. On the other hand, 10% of the observations presents missing values in the second data set. Feelders (1999) concludes that the imputation procedure yields significantly less missclassification error rate.

Surrogate splits were also introduced in the Conditional Inference Forests (Hothorn, Hornik, and Zeileis, 2006) and compared with k nearest neighbor imputation (*knn*−imputation) (Troyanskaya et al., 2001) (implemented in the `R` package `yaImpute` (Crookston and Finley, 2008)) by Rieger, Hothorn, and Strobl (2010) with no clear advantage for either *knn*-imputation or surrogate splits. This study considers classification and regression problems with three different correlation structures and seven schemes for missing values. The comparison between the approaches is measured through the binomial log-likelihood for classification and the mean squared error (MSE) for regression. However, the percentage of missing values is kept constant.

Breiman (2003) proposes an algorithm based on random forests to impute the missing values making use of the proximity matrix. Ishioka (2013) proposes an improvement considering not only the observed values in the imputation procedure, but the nearest neighbors for continuous variables and all the observations for categorical variables. Ishioka (2013) compares these two approaches and *knn*-imputation introducing missing values completely at random in the *Spam* data set (Dua and Graff, 2017) and considering missing data rates from 5% to 60%, concluding that the proposed approach outperforms *knn*-imputation and the previous method proposed by Breiman (2003). However, the same author comments that other mechanisms of missingness should be consider.

Stekhoven and Bühlmann (2011) introduce the missForest algorithm which imputes the missing values iteratively considering it as a regression problem in which the imputation of the current variable is done using all the other variables. In the simulation study presented they consider the classification and regression problems in 7 different data sets where 10%, 20% or 30% of the values are removed completely at random and the performance is asses using the normalized root mean squared error (Oba et al., 2003) for continuous variables and falsely classified entries for categorical variables, concluding that missForest outperforms *knn*-imputation and multivariate imputation chained equations (MICE) (Van Buuren et al., 2006). Furthermore, the authors ensure that the full potential of missForest is deployed when data includes interactions or non-linear relations between variables of unequal scales and different types.

The work of Farhangfar, Kurgan, and Dy (2008) presents a comparison of classification methods like support vector machines, k nearest neighbors and C4.5 applied to missing data with imputation algorithms, including single imputation and MICE. In total, 15 data sets are considered inducing missing values with up to 50% of the observations per variable being set missing. The authors conclude that the application of MICE leads to better results in most of the instances. Based on the previous work of Farhangfar, Kurgan, and Dy (2008) and Rieger, Hothorn, and Strobl (2010), Hapfelmeier, Hothorn, and Ulm (2012) present a comparison study using trees built with the CART criterion, conditional inference trees and their corresponding random forests focusing on surrogate splits to handle missing values, moreover the comparison

also includes the use of MICE to impute missing values. The authors consider 12 real life data sets, half of them for regression and the other half for classification, 8 of the data sets already present missing values while in the rest 4 data sets missing values are induced. Arguing that Rieger, Hothorn, and Strobl (2010) found similar results for MCAR and MAR mechanisms, the missing values are solely introduced according to an MCAR mechanism of missingness, considering missing rates between 0% (benchmark) to 40%. Their results do not show a clear improvement by using multiple imputation, with MICE even producing inferior results when missing values are limited in number and are not arbitrary spread across the data. The results also show a similar result between trees constructed with the CART criterion and conditional inference trees.

The simulation perform by Josse et al. (2019) studies the performance of several methods to handle missing values in regression tasks. In this study 3 different regression functions are consider, one of them being linear, one quadratic and the third being the so-called "friedman1" (Friedman et al., 1991) which has been used as a benchmark for other authors (Friedman et al., 1991; Breiman, 1996; Rieger, Hothorn, and Strobl, 2010). They consider the MCAR mechanism and censoring, inducing up to 20% of observations with missing values in the first variable. The authors compare conditional inference trees, trees and random forests based on the CART criterion and XGBoost (Chen and Guestrin, 2016). The algorithms to handle missing values include Missing Incorporated in Attributes (MIA) (Twala, Jones, and Hand, 2008), surrogate splits for both trees built with the CART criterion and conditional inference trees, block propagation (only implemented in XGBoost), surrogate splits, mean imputation and EM imputation (Dempster, Laird, and Rubin, 1977). The performance metric to compare the methods is the explained variance (i.e. the R^2 statistic). The authors clearly favors the usage of MIA for tree-based methods, while block propagation could also be a good method.

2.2 Simulation Framework

The regression function in this study is the so-called "friedman1" (Friedman et al., 1991), which has been used in previous simulation studies (Breiman, 1996; Rieger, Hothorn, and Strobl, 2010; Josse et al., 2019), given by

$$m(\mathbf{x}) = 10 \sin\left(\pi \mathbf{x}^{(1)} \mathbf{x}^{(2)}\right) + 20\left(\mathbf{x}^{(3)} - 0.5\right)^2 + 10\mathbf{x}^{(4)} + 5\mathbf{x}^{(5)}$$

Following the schema presented by Rieger, Hothorn, and Strobl (2010), we simulate $\mathbf{X}$ as a uniformly distributed variable on $[0,1]^5$ and introduce missing values in $\mathbf{X}^{(1)}, \mathbf{X}^{(3)}$ and $\mathbf{X}^{(4)}$, considering 7 different mechanisms of missingness. For each mechanism of missingness we create one testing data set and 100 training data sets. Each training data set contains 200 observations and the testing data set contains 2000 observations. This amount of data is to have an appropriate approximation to the MSE

$$\mathbb{E}_{\mathbf{X}|\mathcal{D}_n}\left[m_{M,n}(\mathbf{X}) - m(\mathbf{X})\right]^2$$

and the bias

$$\mathbb{E}_{\mathbf{X}|\mathcal{D}_n}\left[m_{M,n}(\mathbf{X}) - m(\mathbf{X})\right].$$

Note that these expressions are conditioned on the training sample $\mathcal{D}_n$ and thus they are random variables which take a different value for each of the 100 training data sets.

In $\mathbf{X}^{(1)}$ 20% of data is missing, in $\mathbf{X}^{(3)}$ the amount is 10%, and in $\mathbf{X}^{(4)}$ there is 20% again. The fraction of missingness is the same in all the training data sets and remains

the same through all the chapter, except for Section 2.4, in which we study the behavior of the distinct approaches changing the amount of missing values. We do not introduce missing values in the testing data set. A random forest is built for each one of the 100 training data sets for the 7 mechanisms of missingness and the data sets without missing values (that are use as benchmark), using $M = 50$ trees, which has been seen by simulation to be sufficient to stabilize the error in the case of complete training data sets. For the remain parameters we use the default values in the regression mode of the `R` package `randomForests`, the parameter `mtry` is set to $\lfloor p/3 \rfloor$, we have sampled without replacement, so a_n is set to $\lceil 0.632n \rceil$, `mtry` is set to 1 and `nodesize` is set to 5. We now describe the mechanisms of missingness.

Missing Completely at Random (MCAR)

We select many locations as desired sampled out of the n observations and replace them by `NA`.

Missing at Random

Methods for generating values missing at random are more complicated. The choice of the locations that are replaced by missing values in the "missing" variable now depends on the value of a second variable, we call this variables "determining" variables as Rieger, Hothorn, and Strobl (2010). Therefore, the values of the "determining" variable now have influence on whether a value in the "missing" variable is missing or not. For $\mathbf{X}^{(1)}$ the "determining" variable is $\mathbf{X}^{(2)}$, while $\mathbf{X}^{(5)}$ is used as the "determining" variable for $\mathbf{X}^{(3)}$ and $\mathbf{X}^{(4)}$.

Creation of ranks (MAR1) The probability for a missing value in a certain location in the "missing" variable is computed by dividing the rank of the location in the "determining" variable by $n(n+1)/2$. The locations for `NA` in the "missing" variable are then sampled with the resulting probability vector.

Creation of two groups (MAR2) We divide the data set in two groups defined by the "determining" variable. A value belongs to the first group if the value in the "determining" variable is greater than or equal to the median of the "determining" variable, otherwise it belongs to the second group. An observation in the respective group has a missing value with probability of 0.9 or 0.1 divided by the number of members in the group. The locations for `NA` in the "missing" variable are then sampled with the resulting probability vector.

Dexter truncation (MAR3) The observations with the biggest values in the "determining" variable have the "missing" variable replaced by `NA` until the desired fraction of `NA` has been achieved.

Symmetric truncation (MAR4) This method is similar to the previous one but we replace by `NA` the values in the "missing" variable in the observations with the biggest and the smallest values in the "determining" variable.

Missing depending on Y (DEPY) The missing values depend on the value of the response, the probability is 0.1 for observations where $Y \geq 13$, otherwise it is 0.4. The

locations for `NA` in the "missing" variable are then sampled with the resulting probability vector.

Not Missing at Random

Logit modelling (LOG) In this method the probability for `NA` no longer depends on a single "determining" variable but in all the other variables. It is modeled as

$$\text{logit}\left(\mathbb{P}\left[\mathbf{M}^{(h)}=1\right]\right)=-0.5+\sum_{\substack{k=1\\k\neq h}}^{5}\mathbf{X}^{(k)}$$

Therefore, the probability of missingness depends on variables with observed values and variables with missing values.

Figure 2.1 shows the MSE for the complete data set and the seven mechanisms of missingness. As expected, the case with complete observations tends to have lowest values for the MSE, moreover we can observe that the MSE tends to have a similar behavior for all the mechanisms of missingness, reasserting previous observations made by other authors (Rieger, Hothorn, and Strobl, 2010; Hapfelmeier, Hothorn, and Ulm, 2012) advocating for the study solely of the MCAR mechanism since other mechanisms of missingness show similar results. Figure 2.2 shows the bias for the same scenarios, the case with complete observations and the MCAR seem symmetric around zero, that is the trees tend to be unbiased for this mechanisms of missingness. However, we observe severe differences for the bias between different mechanisms of missingness. In the MAR3 scenario we observe that the regression function tends to be underestimated. Remember that in this case the missing values are introduce in the observations where the determining variable takes the biggest values, which could explain this bias. For example we do not observe $\mathbf{X}^{(4)}$ in the places where $\mathbf{X}^{(5)}$ takes its biggest values, and by the structure of the regression function this would mean that when the target variable Y take its biggests values corresponds with observations where $\mathbf{X}^{(4)}$ is missing, which seems to be the most important variable (see Figures 2.20 and 2.21). When the missing values are introduce by symmetric truncation (MAR4) we observe that the bias on the estimation tends be center closer to zero. In the case where the observations are introduced depending on the value of the response Y (DEPY), where observations with low values of Y have more easily missing values, we observe that the regression function tends to be overestimated. In MAR1 and MAR2 we observe a less accentuated underestimation of the regression function, even while bigger values on the determining value would generate easier missing values, this could be due to a more homogeneous groups for MAR2. In the NMAR scenario where all the variables influence in similar way the presence of missing values, we observe again a symmetric behavior around zero.

In Figure 2.3 we present a table with the results obtained by Rieger, Hothorn, and Strobl (2010), of especial interest is the row $s3$ which corresponds to the case where the variables are less correlated in their study. We can observe that our precedure can show slightly improvements. However, this comparison must be taken with reserve since Rieger, Hothorn, and Strobl (2010) allowed a few correlation between the variables, while our predictor variables are independent. Moreover, the amount of computation for our proposal increases exponentially, since for each cut there are $2^{n_{miss}}$ possible assignations, which makes questionable the apparent gain.

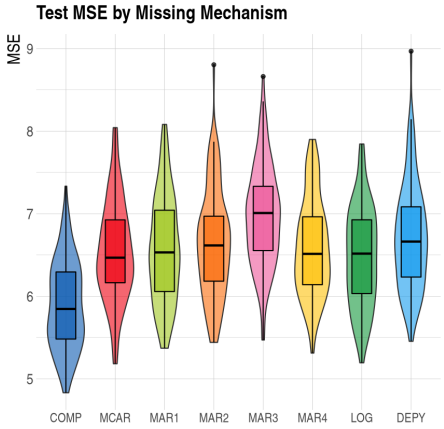

Figure 2.1: MSE of the testing data set for random forests, varying the mechanism of missingness.

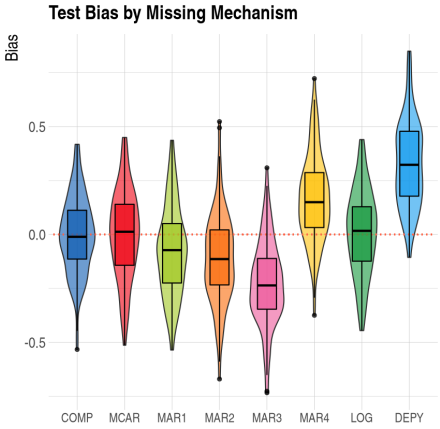

Figure 2.2: Bias of the testing data set for random forests, varying the mechanism of missingness.

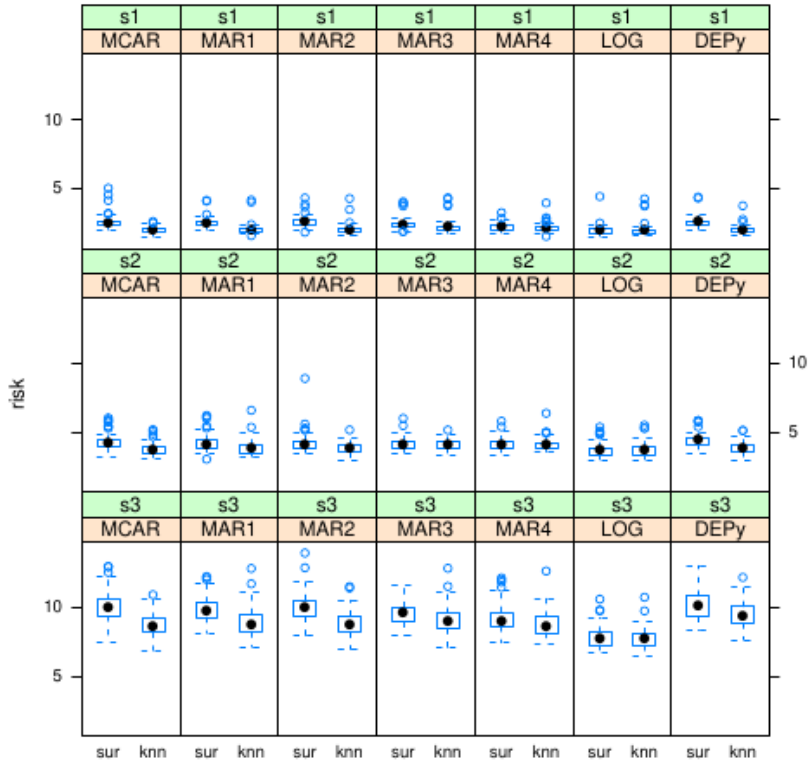

Fig. 3 Distributions of MSE for low dimensional regression problems with missing values in the learning data only.

Figure 2.3: Figure extracted from Rieger, Hothorn, and Strobl (2010), of special interest is the row marked as $s3$ which corresponds to the less correlated variables in their study. The column "sur" corresponds to the use of surrogate splits while the column "knn" correspond to k nearest neighbor imputation.

2.3 Comparison Study

2.3.1 Simple Baselines

As baselines we consider 3 simple methods used in practice to handle missing values, corresponding to:

- Removing the observations with missing values.
- Removing the columns that have missing values.
- Imputing the missing values with the median of the observations in the corresponding variable.

After we have applied each of these options we construct random forests with the usual CART criterion. The parameters of the random forests are the same as in our proposed algorithm, except when we eliminate observations with missing values, in this case we have established $a_n = \lceil 0.632 n_{comp} \rceil$ where n_{comp} is the number of complete observations, the rest of the parameters remain the same.

Figure 2.4 shows the MSE when we decide to eliminate all the incomplete observations (listwise deletion). In this case the scenarios with missing values use less information than the complete one, since at least 20% of the observations has some missing value, in fact on average just $(1 - (1 - 0.1)(1 - 0.2)(1 - 01)) \times 100 = 64.8\%$ of the observations are complete. Figure 2.5 shows the bias eliminating incomplete observations. We can observe that the tendency to underestimate the regression function for the MAR1, MAR2 and MAR3 has been intensified as well as the overestimation of the regression function for the DEPY mechanism of missingness, in fact in this case we did not observe any point underestimating the bias (all the violin plot is above the zero-line). Figures 2.6 and 2.7 show the MSE and the bias when we delete the variables with missing values. Therefore the random forests are built solely with 2 variables, in this case there is no effect of the mechanism of missingness so the MSE and the bias have the same behavior for all the mechanisms. Since the trees are unbiased we observe in the violin plots for the bias of the random forests to be symmetric around zero. However, in terms of the MSE this method shows a poor performance compare to listwise elimination or the use of our algorithm. Evidently this method should be avoid, unless we count with many features with complete information so the elimination of just a few percentage of the features would not be so harmful. Figures 2.8 and 2.9 show the MSE and the bias when we impute the missing values with the median of the observed values for the respective variable. Since the variables with missing values have at least 80% of their information, this option seems reasonable, which is reflected in the values observed for both the MSE and the bias. Between the three simple approaches we conclude that it is the option with the best performance in terms of the MSE.

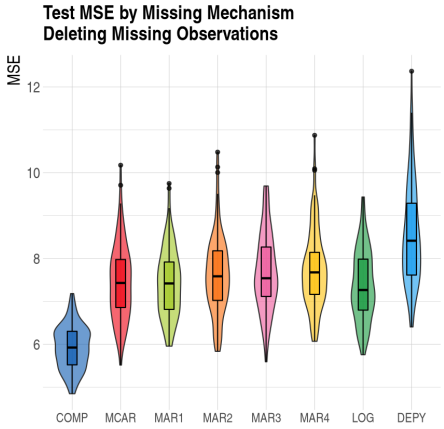

Figure 2.4: MSE of the testing data set for random forests deleting observations with missing values, varying the mechanism of missingness.

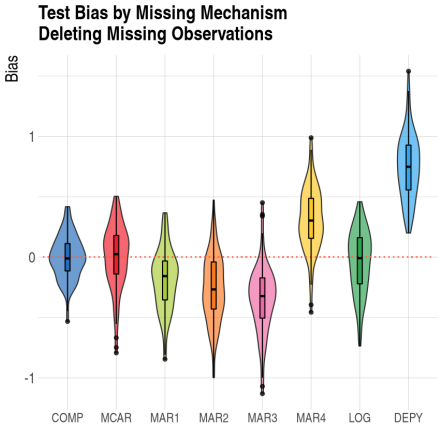

Figure 2.5: Bias of the testing data set for random forests deleting observations with missing values, varying the mechanism of missingness.

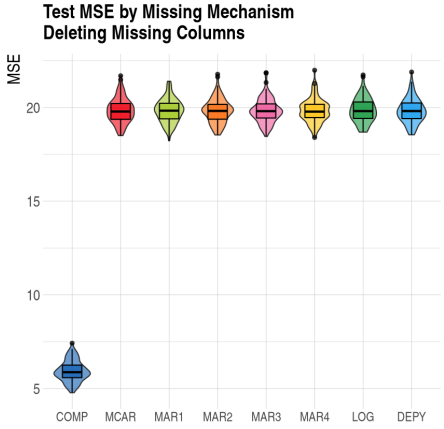

Figure 2.6: MSE of the testing data set for random forests deleting columns with missing values, varying the mechanism of missingness.

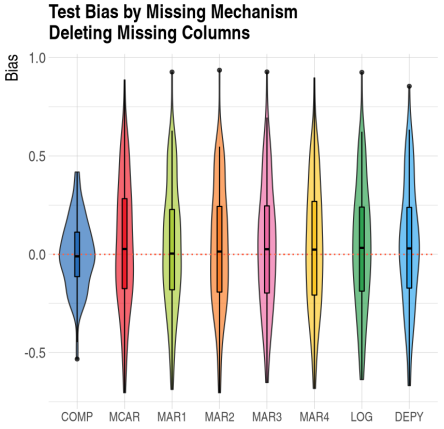

Figure 2.7: Bias of the testing data set for random forests deleting columns with missing values, varying the mechanism of missingness.

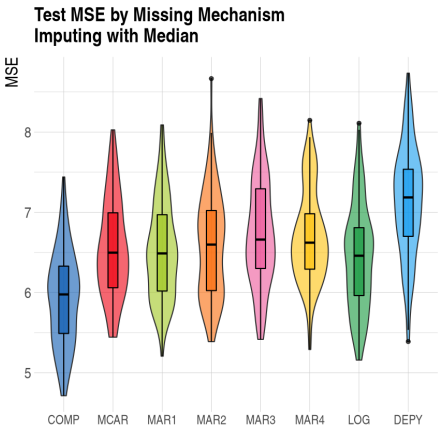

Figure 2.8: MSE of the testing data set for random forests imputing with the median of the observed values, varying the mechanism of missingness.

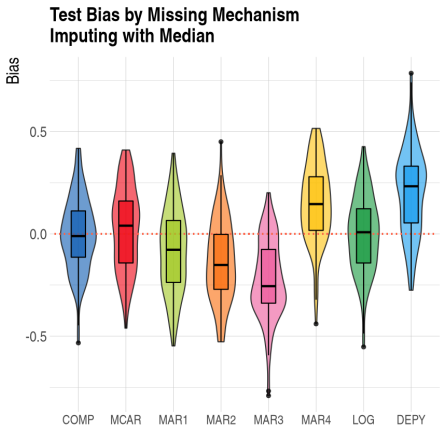

Figure 2.9: Bias of the testing data set for random forests imputing with the median of the observed values, varying the mechanism of missingness.

2.3.2 Imputation Approaches Based on Random Forests

In this section we study three different approaches which impute missing values through random forests. These methods were described in Section 1.6 corresponding to the algorithms presented by Breiman (2003), Ishioka (2013), and Stekhoven and Bühlmann (2011), that we have denoted as Breiman's approach, Ishioka's approach and missForest approach, respectively. These algorithms impute the missing values through iterative improvements, Breiman's approach and Ishioka's approach construct a random forest with the current imputations and use the proximity matrix to update them. Denoting as $K_{M,\ell}(i,j)$ the i,j element of this matrix, then Breiman's updating of the imputation values is given by the formula

$$\widehat{\mathbf{X}}^{(h)}_{j,\ell+1} = \frac{\sum_{i\in \mathbf{i}^{(h)}_{obs}} K_{M,\ell}(i,j)\mathbf{X}^{(h)}_i}{\sum_{i\in \mathbf{i}^{(h)}_{obs}} K_{M,\ell}(i,j)}, \qquad \begin{array}{l}\ell \geq 1 \\ j \in \mathbf{i}^{(h)}_{miss}\end{array}$$

Instead of considering only the observed values to update the imputation, Ishioka's approach considers the k nearest neighbors to improve the imputation, hence the imputation is done according to the formula

$$\widehat{\mathbf{X}}^{(h)}_{j,\ell+1} = \frac{\sum_{\substack{i\in \text{neigh}_k \\ i\neq j}} K_{M,\ell}(i,j)\widehat{\mathbf{X}}^{(h)}_{i,\ell}}{\sum_{\substack{i\in \text{neigh}_k \\ i\neq j}} K_{M,\ell}(i,j)}, \qquad \begin{array}{l}\ell \geq 1 \\ j \in \mathbf{i}^{(h)}_{miss}\end{array}$$

The missForest algorithm treats the imputation problem as a regression by itself, where each variable is imputed with the previous imputation of all the others through the construction of a random forest.

We initialize the algorithms with the median imputation, which accordingly to the previous observations seems to be an appropriate simple approach in terms of the MSE, the rest of the parameters correspond to the default values of the package `missForest` in `R` which are set to 10 iterations and the construction of random forests with 100 trees in each iteration to construct the approximation matrix.

Figures 2.10, 2.12 and 2.14 show the MSE for these algorithms while Figures 2.11, 2.13 and 2.15 show their bias. In terms of the MSE there is no clear advantage of one method over the others, except for Ishioka's approach were it is possible to observe slightly higher values for some mechanisms of missingness like DEPY or MAR2. These methods also seem to be more unbiased, since even mechanisms of missingness where the regression function was underestimated (like MAR3) or overestimated (like DEPY) appear to be more symmetric around zero for the three methods. The fact that in practice the mechanism of missingness is unknown makes desirable these kind of methods that seem to be robust to the missing value generating process in both the MSE and the bias. Comparing these results with the proposed approach in this work we again do not observe a clear advantage in a particular method in terms of the MSE, which could make us wonder if it is really worthy the exhaustive searching for the best assignation of the missing values.

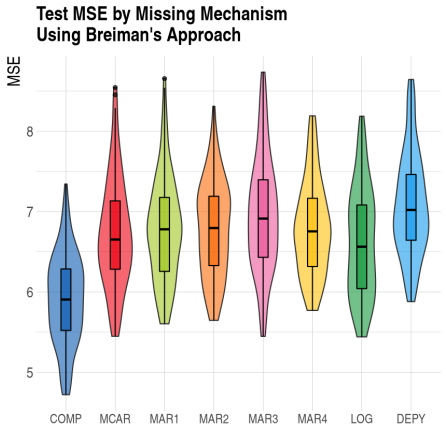

Figure 2.10: MSE of the testing data set for random forests using the imputation procedure proposed by Breiman, varying the mechanism of missingness.

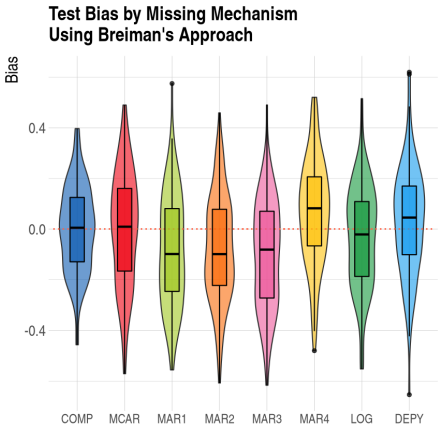

Figure 2.11: Bias of the testing data set for random forests using the imputation procedure proposed by Breiman, varying the mechanism of missingness.

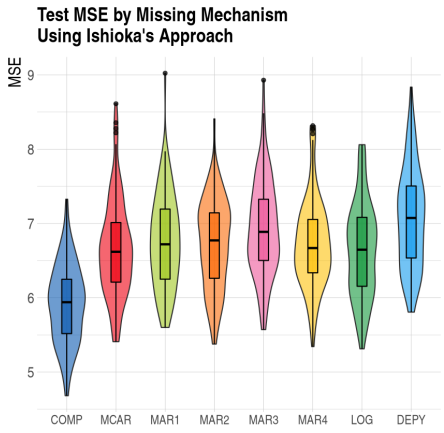

Figure 2.12: MSE of the testing data set for random forests using the imputation procedure proposed by Ishioka, varying the mechanism of missingness.

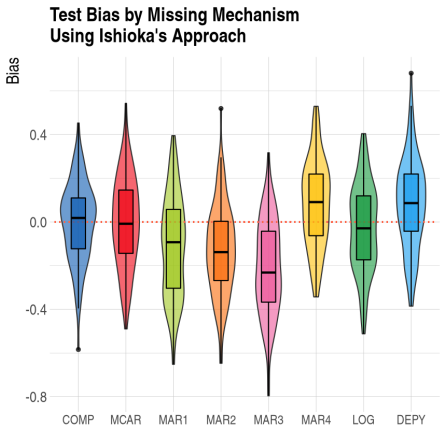

Figure 2.13: Bias of the testing data set for random forests using the imputation procedure proposed by Ishioka, varying the mechanism of missingness.

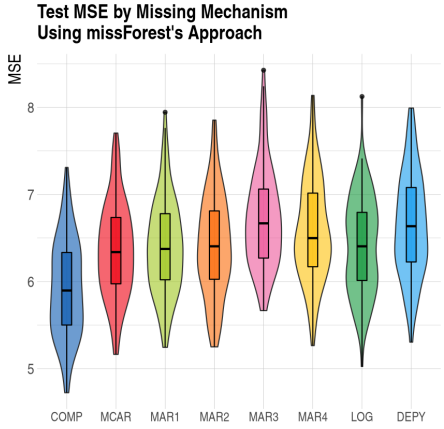

Figure 2.14: MSE of the testing data set for random forests using the missForest's imputation, varying the mechanism of missingness.

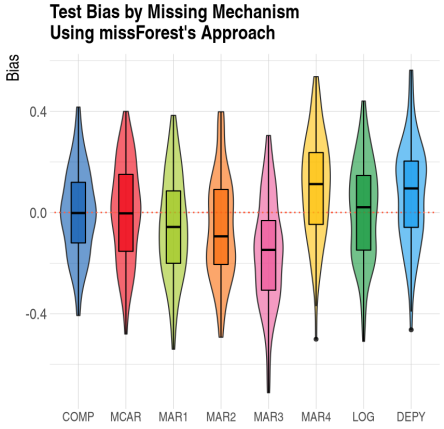

Figure 2.15: Bias of the testing data set for random forests using the missForest's imputation, varying the mechanism of missingness.

2.3.3 Missing Incorporated in Attributes

Missing Incorporated in Attributes (MIA) (Twala, Jones, and Hand, 2008) can be seen as a particular case of our procedure in which observations with missing values are assigned together to the child node that maximize the CART criterion (or any other criterion consider), hence we study this algorithm in this section. We use the same parameters in this case as those used in our approach. Figure 2.16 shows the MSE for this algorithm and Figure 2.17 shows its bias. In terms of the MSE seems to be as robust to the missing mechanisms as the tree-based imputation methods studied in the previous section. However, this robustness to the mechanism of missingness is not

presented anymore when the bias is studied, we can observe for example that the MAR3 and DEPY seem to be highly biased.

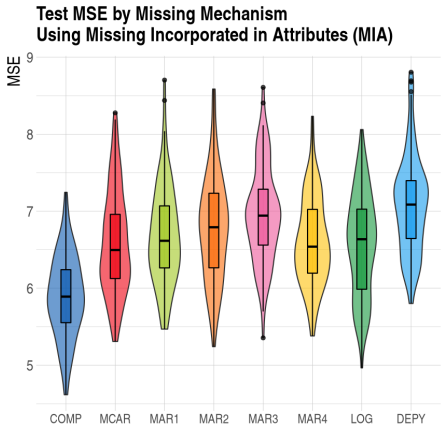

Figure 2.16: MSE of the testing data set for random forests using Missing Incorporated in Attributes, varying the mechanism of missingness.

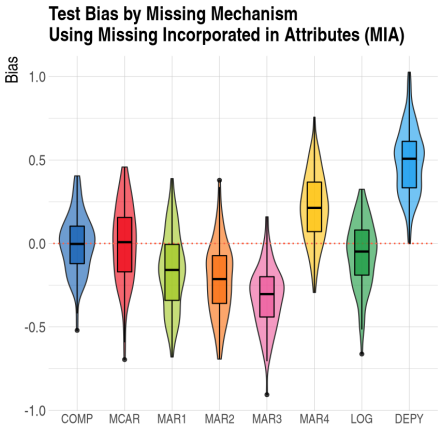

Figure 2.17: Bias of the testing data set for random forests using Missing Incorporated in Attributes, varying the mechanism of missingness.

Figures 2.18 and 2.19 present the average MSE and average Bias for each of the approaches considered, excluding the method of elimination of features with missing values. Listwise deletion generates the largest errors and the estimates with more bias for all the mechanisms. Hence, it could be taken as a an bound of the minimum expected behavior for a method that attempts to estimate the regression function with missing values. We observe that missForest consistently generates estimators with the lowest errors regardless of the missing mechanism. We can observe that our proposed method outperforms MIA, Breiman's approach and Ishioka's approach and can achieve similar errors as missForest (see MAR4, LOG and DEPY). It is worthy to observe that even a simple approach as imputing with the median can outperforms most of the methods considered or with similar behavior in several mechanisms of missingness, only surpassed by missForest (see MAR1, MAR2, MAR3, LOG). Moreover it can achieve a similar performance as a computationally expensive procedure as the one proposed in this work. In terms of bias, we observe that the algorithms that impute the missing values before the construction of the random forests tend to generate more unbiased results, while MIA tends to generate the second more biased estimators (just upper listwise deletion), our proposal can achieve similar behavior to imputation methods (see MAR1, MAR2, LOG) or being somewhere between MIA and median imputation (see MAR3, MAR4, DEPY). For the MCAR case, all the methods considered tend to be unbiased and with similar MSE, except for missForest and listwise deletion with the lowest and highest MSE, respectively.

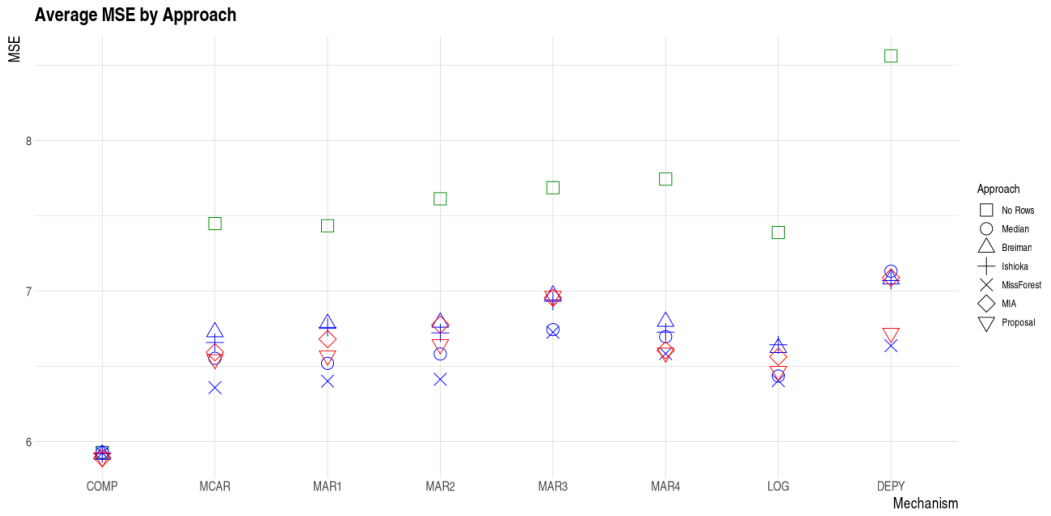

Figure 2.18: Average MSE of the testing data set for each approach (excluding deletion of columns with missing values) and each mechanism of missingness. In green there is listwise deletion, those apporaches that implement some imputation in the data set are in blue(median imputation, Breiman's approach, Ishioka's approach and missForest) and those apporaches that handle missing values directly in the learning step are in red (MIA, our proposal).

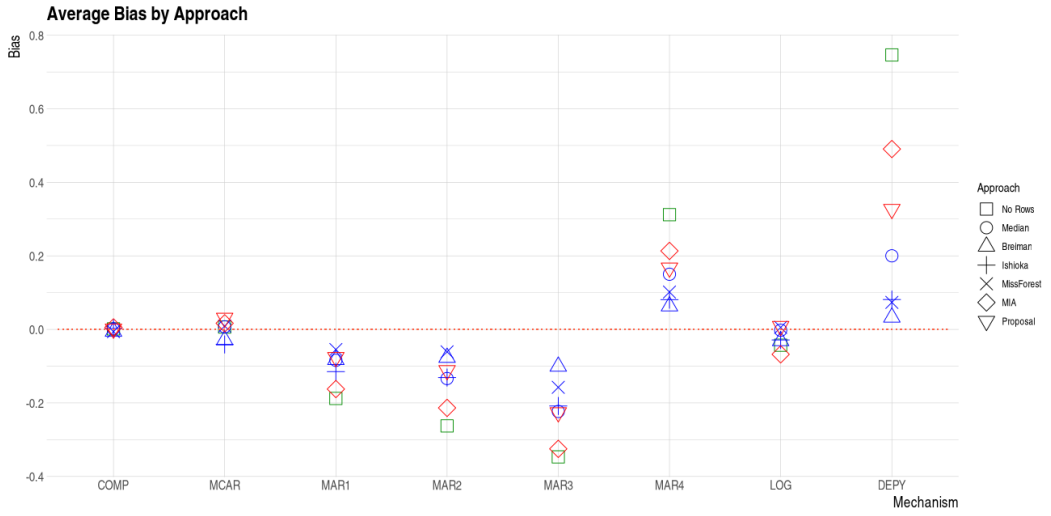

Figure 2.19: Average Bias of the testing data set for each approach (excluding deletion of columns with missing values) and each mechanism of missingness. In green there is listwise deletion, those apporaches that implement some imputation in the data set are in blue(median imputation, Breiman's approach, Ishioka's approach and missForest) and those apporaches that handle missing values directly in the learning step are in red (MIA, our proposal).

2.4 Varying the Rate of Missing Values

Using the 100 training data sets with no missing values, we calculate the importance of the variables with the `R` package `randomForests`, by percentage of increase in mean squared error and by increase in node purity (Breiman, 2001; Breiman, 2003). Figures 2.20 and 2.21 show the violin plots for the mean squared error and the increase in node purity, respectively. We can see a consistently order for the variables with missing values in both measures of importance, where $\mathbf{X}^{(4)}$ is consider more important than $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(3)}$. Hence, we decided to change the fraction of missingness in $\mathbf{X}^{(4)}$ to vary between 5%, 10%, 20%, 40%, 60%, 80%, 90% and 95%, without changing the percentage of missingness for $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(3)}$. Figure 2.22 shows the MSE , we observe than even for large fraction of missing values the algorithm estimate the regression function with little deterioration in terms of the MSE, furthermore the method seems to be robust to the missing mechanism. Figure 2.23 shows the bias, when the missing values are introduce completely at random (MCAR) or when they are introduce in the observations corresponding to the biggest and lowest values of the "determining" variable (MAR4), the estimation appears to be unbiased. The NMAR case where the missing values are introduced according to a logit model that considers all the other variables also seems to be create unbiased estimators. For the rest of the mechanisms of missingness it is clear that the estimations are biased, and this effect tends to increase with the number of observations with missing values. The scenario were the missing values are introduced in the places where the "determining" variable takes its biggest values (MAR3) as well as the case were the missing values are introduced depending on the value of the response Y are the most affected by this biased effect.

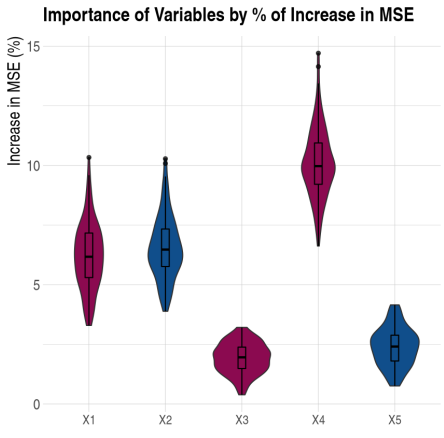

Figure 2.20: Importance Variable accordingly to percentage increase in MSE.

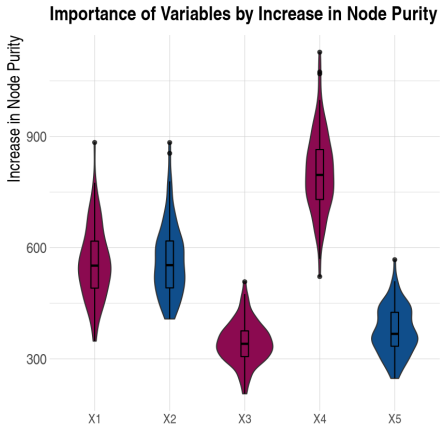

Figure 2.21: Importance Variable accordingly to increase in node purity.

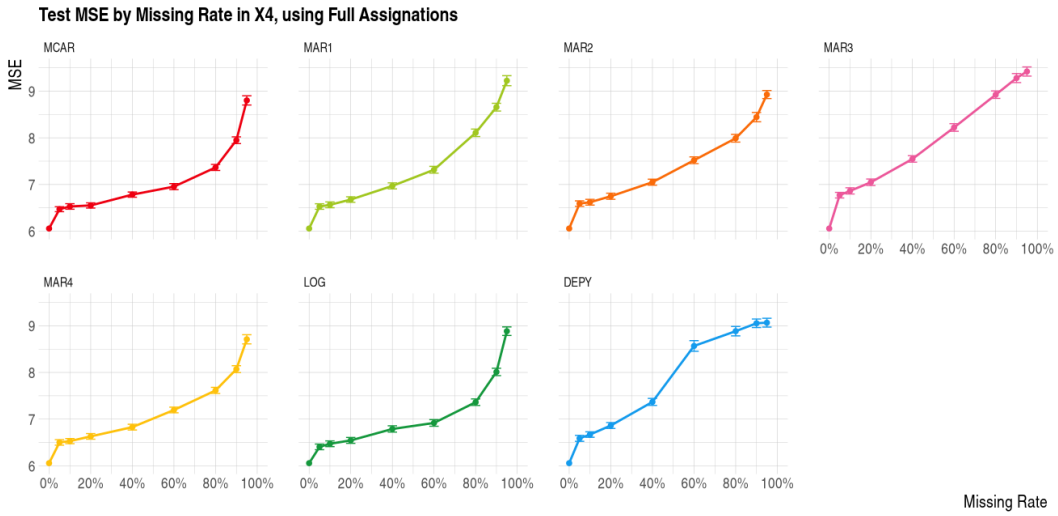

Figure 2.22: MSE of the test data set, varying the percentage of missing values in $\mathbf{X}^{(4)}$, missing values percentage for $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(3)}$ are set to 20% and 10%, respectively.

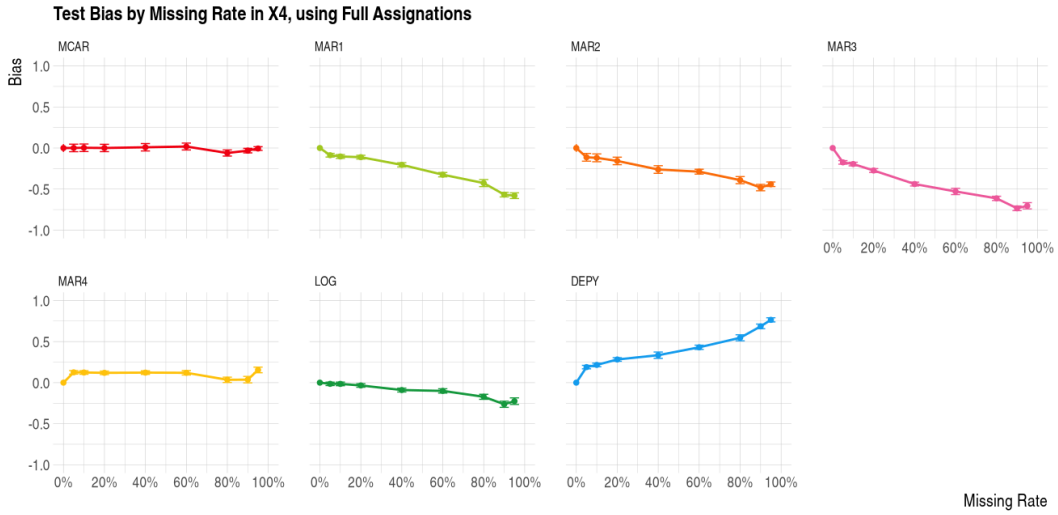

Figure 2.23: Bias of the test data set, varying the percentage of missing values in $\mathbf{X}^{(4)}$, missing values percentage for $\mathbf{X}^{(1)}$ and $\mathbf{X}^{(3)}$ are set to 20% and 10%, respectively.

We have also studied the MSE and bias of the distinct methods considered here, varying the percentage of missing values for the seven mechanisms of missingness. Tables 2.1 and 2.2 show the MSE for the MCAR case, and Tables 2.3 and 2.4 show the bias. The tables with the results for the rest of the mechanisms of missingness are found in Appendix B. In general, we observe that missForest tends to outperform the rest of algorithms regardless of the mechanism of missingness and the percentage of missing values. However, important differences between the performance of the methods become clear with the increasing percentage of missing values. Especially when the percentage of observations presenting missing values is over 60% there is an order

on the methods, induced by the MSE. In this cases missForest, our proposal and MIA represent the methods with less MSE. Moreover, we can see the advantage of searching for the best assignation when the percentage arises between 90% and 95% with our proposal outperforming all the other algorithms (in the case of DEPY this phenomenon is achieved even for 80%).

The differences between the mechanisms of missingness are also reflected when we increase the percentage of missing values. For example, the top three algorithms (missForest, our approach and MIA) present a MSE between 7.95 and 8.19 when the percentage of missingness is 90% and the introduction of missing values is done completely at random (MCAR). On the hand, for the same percentage of missing values and the same algorithms we observe a deterioration in terms of the MSE which varies between 9.05 and 12.86 when we consider the DEPY case. Actually , for the DEPY case we consistently see that our approach and missForest outperform the other methods, regardless of the percentage of missing values, while there is no clear advantage of the rest of the algorithms over the others (excluding listwise deletion and deletion of variables).

When we include the bias in the study, the differences between methods and data-missing mechanisms become more evident. For instance, we observe an unbiased behavior for the MCAR and the LOG cases, with some deterioration when the percentage of missing values affects up to 60% of the observations. Conversely, for several mechanisms we observe a severe bias for most of the methods (see MAR1, MAR2, MAR3 and DEPY). Missing-imputation techniques like Breiman's approach, Ishioka's approach or missForest seem to be the less affected in terms of the bias. While median imputation, MIA and listwise deletion are the methods getting more biased estimators. For small values in the percentage of missingness our approach also presents drawbacks compare to other techniques in terms of the bias, however these issues are overcome as the percentage of missingness increases.

In conclusion, we observe that the differences between distinct techniques and data-missing mechanisms become clear when the percentage of observations with some missing values increases, especially when this value is over 60%. While these differences are diluted for small values of this percentage (less than 40%). Computer exhaustive algorithms seem to perform particularly well for large percentage of missing values. Moreover the behavior of the methods appear to be dependent on the mechanim of missingness. Moreover, important differences between the algorithms can be observed when we consider the bias.

Approach	0	5	10	20	40
No Rows	6.06 ± 0.06	6.85 ± 0.07	7.05 ± 0.07	7.47 ± 0.08	8.38 ± 0.09
No Columns	6.06 ± 0.06	19.76 ± 0.06	19.78 ± 0.06	19.80 ± 0.06	19.76 ± 0.06
Median	6.06 ± 0.06	6.40 ± 0.06	6.50 ± 0.06	6.66 ± 0.06	7.20 ± 0.06
Breiman	6.06 ± 0.06	6.42 ± 0.06	6.51 ± 0.06	6.71 ± 0.06	7.06 ± 0.07
Ishioka	6.06 ± 0.06	6.31 ± 0.06	6.41 ± 0.06	6.64 ± 0.06	6.88 ± 0.06
MissForest	6.06 ± 0.06	6.43 ± 0.06	6.42 ± 0.05	6.45 ± 0.06	6.53 ± 0.05
MIA	6.06 ± 0.06	6.29 ± 0.06	6.35 ± 0.06	6.55 ± 0.06	6.78 ± 0.06
Proposal	6.06 ± 0.06	6.47 ± 0.05	6.53 ± 0.06	6.55 ± 0.06	6.78 ± 0.06

Table 2.1: Average mean squared error and its standard error for the different methods, considering the MCAR case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	60	80	90	95
No Rows	9.80 ± 0.15	12.88 ± 0.20	17.60 ± 0.43	22.70 ± 0.55
No Columns	19.77 ± 0.06	19.77 ± 0.06	19.78 ± 0.06	19.78 ± 0.06
Median	8.14 ± 0.09	10.20 ± 0.14	12.36 ± 0.22	13.16 ± 0.22
Breiman	7.53 ± 0.08	8.83 ± 0.13	10.63 ± 0.23	11.96 ± 0.28
Ishioka	7.32 ± 0.07	8.03 ± 0.09	9.17 ± 0.14	10.45 ± 0.19
MissForest	6.80 ± 0.07	7.26 ± 0.07	8.19 ± 0.13	9.48 ± 0.32
MIA	7.15 ± 0.08	7.75 ± 0.09	8.79 ± 0.14	10.08 ± 0.20
Proposal	6.96 ± 0.07	7.37 ± 0.06	7.95 ± 0.07	8.80 ± 0.10

Table 2.2: (Cont.). Average mean squared error and its standard error for the different methods, considering the MCAR case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	0	5	10	20	40
No Rows	0.00 ± 0.02	-0.01 ± 0.02	0.02 ± 0.02	-0.01 ± 0.03	-0.02 ± 0.03
No Columns	0.00 ± 0.02	0.01 ± 0.03	0.01 ± 0.03	0.01 ± 0.03	0.01 ± 0.03
Median	0.00 ± 0.02	-0.01 ± 0.02	-0.01 ± 0.02	-0.01 ± 0.02	0.01 ± 0.03
Breiman	0.00 ± 0.02	0.01 ± 0.02	-0.03 ± 0.02	-0.04 ± 0.02	-0.05 ± 0.03
Ishioka	0.00 ± 0.02	-0.05 ± 0.02	-0.05 ± 0.02	-0.04 ± 0.02	-0.03 ± 0.02
MissForest	0.00 ± 0.02	0.00 ± 0.02	-0.02 ± 0.02	-0.01 ± 0.02	-0.01 ± 0.02
MIA	0.00 ± 0.02	0.00 ± 0.02	-0.02 ± 0.02	0.00 ± 0.02	0.00 ± 0.02
Proposal	0.00 ± 0.02	0.00 ± 0.02	0.00 ± 0.02	0.00 ± 0.02	0.01 ± 0.02

Table 2.3: Average bias and its standard error for the different methods, considering the MCAR case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

Approach	60	80	90	95
No Rows	0.03 ± 0.04	-0.02 ± 0.06	-0.09 ± 0.11	0.09 ± 0.18
No Columns	0.00 ± 0.03	0.00 ± 0.03	-0.02 ± 0.03	-0.01 ± 0.03
Median	-0.02 ± 0.03	-0.05 ± 0.03	-0.04 ± 0.05	-0.07 ± 0.06
Breiman	-0.07 ± 0.03	-0.08 ± 0.03	-0.10 ± 0.03	-0.10 ± 0.04
Ishioka	-0.05 ± 0.03	-0.08 ± 0.03	-0.09 ± 0.04	-0.12 ± 0.05
MissForest	-0.01 ± 0.03	-0.01 ± 0.04	0.07 ± 0.05	-0.08 ± 0.07
MIA	-0.01 ± 0.03	-0.05 ± 0.03	-0.07 ± 0.05	-0.09 ± 0.07
Proposal	0.02 ± 0.03	-0.06 ± 0.03	-0.03 ± 0.04	-0.01 ± 0.05

Table 2.4: (Cont). Average bias and its standard error for the different methods, considering the MCAR case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

2.5 Decomposition of the CART Criterion

Applying elementary algebra (see Appendix A), we can show that our CART criterion might be written as

$$L_n\left(A,d,w\right) = L_{1,n}(A,d) + L_{2,n}\left(A,d,w\right) + L_{3,n}\left(A,d,w\right) + L_{4,n}\left(A,d,w\right)$$

where

$$\begin{aligned} L_{1,n}(A,d) =& \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A,obs}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, \mathbf{M}_i^{(h)}=0} \\ & - \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_L,obs}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, a^{(h)} \leq \mathbf{X}_i^{(h)} \leq z,\, \mathbf{M}_i^{(h)}=0} \\ & - \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_R,obs}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, z \leq \mathbf{X}_i^{(h)} \leq b^{(h)},\, \mathbf{M}_i^{(h)}=0} \end{aligned}$$

$$\begin{aligned} L_{2,n}(A,d,w) =& \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A,miss}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, \mathbf{M}_i^{(h)}=1} \\ & - \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_L,miss}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, a^{(h)} \leq \widehat{\mathbf{X}}_{i,out}^{(h)} \leq z,\, \mathbf{M}_i^{(h)}=1} \\ & - \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_R,miss}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in} \in A,\, z \leq \widehat{\mathbf{X}}_{i,out}^{(h)} \leq b^{(h)},\, \mathbf{M}_i^{(h)}=1} \end{aligned}$$

$$\begin{aligned} L_{3,n}(A,d,w) =& \frac{\widehat{N}_{obs}(A)}{\widehat{N}(A)} \left(\widehat{Y}_{A,obs} - \widehat{Y}_A\right)^2 \\ & - \frac{\widehat{N}_{obs}^{(h)}(A_L)}{\widehat{N}(A)} \left(\widehat{Y}_{A_L,obs} - \widehat{Y}_{A_L}\right)^2 \\ & - \frac{\widehat{N}_{obs}^{(h)}(A_R)}{\widehat{N}(A)} \left(\widehat{Y}_{A_R,obs} - \widehat{Y}_{A_R}\right)^2 \end{aligned}$$

$$\begin{aligned} L_{4,n}(A,d,w) =& \frac{\widehat{N}_{miss}(A)}{\widehat{N}(A)} \left(\widehat{Y}_{A,miss} - \widehat{Y}_A\right)^2 \\ & - \frac{\widehat{N}_{miss}^{(h)}(A_L)}{\widehat{N}(A)} \left(\widehat{Y}_{A_L,miss} - \widehat{Y}_{A_L}\right)^2 \\ & - \frac{\widehat{N}_{miss}^{(h)}(A_R)}{\widehat{N}(A)} \left(\widehat{Y}_{A_R,miss} - \widehat{Y}_{A_R}\right)^2 \end{aligned}$$

Analogously to equation (1.1) it can be shown that

$$L_{1,n}(A,d) = \left(\frac{\widehat{N}_{obs}^{(h)}(A)}{\widehat{N}(A)}\right) \left(\frac{\widehat{N}_{obs}^{(h)}(A_L)\widehat{N}_{obs}^{(h)}(A_R)}{\widehat{N}_{obs}(A)\widehat{N}_{obs}(A)}\right) \left(\widehat{Y}_{A_L,obs} - \widehat{Y}_{A_R,obs}\right)^2 \tag{2.1}$$

and

$$L_{2,n}(A,d,w) = \left(\frac{\widehat{N}^{(h)}_{miss}(A)}{\widehat{N}(A)}\right)\left(\frac{\widehat{N}^{(h)}_{miss}(A_L)\widehat{N}^{(h)}_{miss}(A_R)}{\widehat{N}_{miss}(A)\widehat{N}_{miss}(A)}\right)\left(\widehat{Y}_{A_L,miss} - \widehat{Y}_{A_R,miss}\right)^2 \quad (2.2)$$

Let be

- $p_L = \mathbb{P}\left[a^{(h)} \leq \mathbf{X}^{(h)} < z | \mathbf{X} \in A, \mathbf{M}^{(h)} = 0\right]$
- $\mu_A = \mathbb{E}[Y|\widehat{\mathbf{X}}_{in} \in A]$
- $\mu_{A,obs} = \mathbb{E}[Y|\widehat{\mathbf{X}}_{in} \in A, \mathbf{M}^{(h)} = 0]$
- $\mu_{A,miss} = \mathbb{E}[Y|\widehat{\mathbf{X}}_{in} \in A, \mathbf{M}^{(h)} = 1]$
- $\mu_{A_L} = \mathbb{E}[Y|\widehat{\mathbf{X}}_{in} \in A, a^{(h)} \leq \widehat{\mathbf{X}}_{out} < z]$ (resp. μ_{A_R})
- $\mu_{A_L,obs} = \mathbb{E}[Y|\widehat{\mathbf{X}}_{in} \in A, a^{(h)} \leq \widehat{\mathbf{X}}_{out} < z, \mathbf{M}^{(h)} = 0]$ (resp. $\mu_{A_R,obs}$)
- $\mu_{A_L,miss} = \mathbb{E}[Y|\widehat{\mathbf{X}}_{in} \in A, a^{(h)} \leq \widehat{\mathbf{X}}_{out} < z, \mathbf{M}^{(h)} = 1]$ (resp. $\mu_{A_R,miss}$)

then, the theoretical CART criterion can be written as $L^\star(A,d,w) = L_1^\star(A,d) + L_2^\star(A,d,w) + L_3^\star(A,d,w) + L_4^\star(A,d,w)$, where

$$\begin{aligned} L_1^\star(A,d) &= \mathbb{V}\left[Y|\widehat{\mathbf{X}}_{in} \in A,\, \mathbf{M}^{(h)} = 0\right]\mathbb{P}\left[\mathbf{M}^{(h)} = 0|\widehat{\mathbf{X}}_{in} \in A\right] \\ &- \mathbb{V}\left[Y|\widehat{\mathbf{X}}_{in} \in A,\, a^{(h)} \leq \mathbf{X}^{(h)} < z,\, \mathbf{M}^{(h)} = 0\right]\mathbb{P}\left[\mathbf{M}^{(h)} = 0|\widehat{\mathbf{X}}_{in} \in A\right]p_L \\ &- \mathbb{V}\left[Y|\widehat{\mathbf{X}}_{in} \in A,\, z \leq \mathbf{X}^{(h)} \leq b^{(h)},\, \mathbf{M}^{(h)} = 0\right]\mathbb{P}\left[\mathbf{M}^{(h)} = 0|\widehat{\mathbf{X}}_{in} \in A\right](1-p_L) \end{aligned}$$

$$\begin{aligned} L_2^\star(A,d,w) &= \mathbb{V}\left[Y|\widehat{\mathbf{X}}_{in} \in A,\, \mathbf{M}^{(h)} = 1\right]\mathbb{P}\left[\mathbf{M}^{(h)} = 1|\widehat{\mathbf{X}}_{in} \in A\right] \\ &- \mathbb{V}\left[Y|\widehat{\mathbf{X}}_{in} \in A,\, a^{(h)} \leq \widehat{\mathbf{X}}^{(h)}_{out} < z,\, \mathbf{M}^{(h)} = 1\right]\mathbb{P}\left[\mathbf{M}^{(h)} = 1|\widehat{\mathbf{X}}_{in} \in A\right]w(Y) \\ &- \mathbb{V}\left[Y|\widehat{\mathbf{X}}_{in} \in A,\, z \leq \widehat{\mathbf{X}}^{(h)}_{out} \leq b^{(h)},\, \mathbf{M}^{(h)} = 1\right]\mathbb{P}\left[\mathbf{M}^{(h)} = 1|\widehat{\mathbf{X}}_{in} \in A\right](1-w(Y)) \end{aligned}$$

$$\begin{aligned} L_3^\star(A,d,w) = &\left(\mu_{A,obs} - \mu_A\right)^2\mathbb{P}\left[\mathbf{M}^{(h)} = 0|\widehat{\mathbf{X}}_{in} \in A\right] \\ &- \left(\mu_{A_L,obs} - \mu_{A_L}\right)^2\mathbb{P}\left[\mathbf{M}^{(h)} = 0|\widehat{\mathbf{X}}_{in} \in A\right]p_L \\ &- \left(\mu_{A_R,obs} - \mu_{A_R}\right)^2\mathbb{P}\left[\mathbf{M}^{(h)} = 0|\widehat{\mathbf{X}}_{in} \in A\right](1-p_L) \end{aligned}$$

$$\begin{aligned} L_4^\star(A,d,w) = &\left(\mu_{A,miss} - \mu_A\right)^2\mathbb{P}\left[\mathbf{M}^{(h)} = 1|\widehat{\mathbf{X}}_{in} \in A\right] \\ &- \left(\mu_{A_L,miss} - \mu_{A_L}\right)^2\mathbb{P}\left[\mathbf{M}^{(h)} = 1|\widehat{\mathbf{X}}_{in} \in A\right]w(Y) \\ &- \left(\mu_{A_R,miss} - \mu_{A_R}\right)^2\mathbb{P}\left[\mathbf{M}^{(h)} = 1|\widehat{\mathbf{X}}_{in} \in A\right](1-w(Y)) \end{aligned}$$

We can give an interpretation to $L_1^\star$, $L_2^\star$, $L_3^\star$ and $L_4^\star$. It is clear that $L_1^\star$ (resp. $L_2^\star$) measures the change of variance of the points where the split variable is observed (missing) when a cut is performed, while $L_3^\star$ (resp. $L_4^\star$) looks to measure the change of the

squared bias of the points where the split variable is observed (missing), leading to the well-known bias-variance trade-off. Thus, when $L_3^\star$ or $L_4^\star$ are different from zero we can conclude that the missing mechanism is introducing a source of bias into the estimation of the regression function. We have observed in our simulations that the MCAR mechanism seems to not introduce any bias in the estimation of the regression function. Thus, we expect that $L_{3,n}$ and $L_{4,n}$ would take values near zero. Our simulation shows results that sustain these observations. However further work is needed and a proof remains necessary.

Figure 2.24 shows the value of the CART criterion as well as its component parts for each cell of a tree constructed with a data set where the missing values were introduced completely at random, 10% for $\mathbf{X}^{(1)}$, 20% for $\mathbf{X}^{(3)}$ and 10% for $\mathbf{X}^{(4)}$. Figure 2.25 shows the boxplots of these values. For each tree in a random forests we can take the mean value for these function and create a boxplot with the latter, in this case we would appreciate the behavior of the component parts for the entire forest form by $M = 50$ trees, Figure 2.26 shows these boxplots where we can appreciate how $L_{3,n}$ and $L_{4,n}$ take values near zero. We observe also that $L_{2,n}$ take smaller values than $L_{1,n}$, however this could be due to the fact that the amount observations with missing values is less than the opposite, which would make $L_{1,n}$ to gain more importance in the CART criterion. We construct a random forest for each one of the 100 data sets with missing values introduced completely at random, we can take the mean value of $L_{1,n}$, $L_{2,n}$, $L_{3,n}$ and $L_{4,n}$ and construct the corresponding boxplots, which will show the behavior of these functions for the MCAR mechanisms. These boxplots are presented in Figure 2.27 where we can observe that $L_{3,n}$ and $L_{4,n}$ tends to take values near zero for the MCAR mechanism, while Figures 2.2 and 2.23 shows that under this mechanism of missingness the estimations of the regression functions are unbiased.

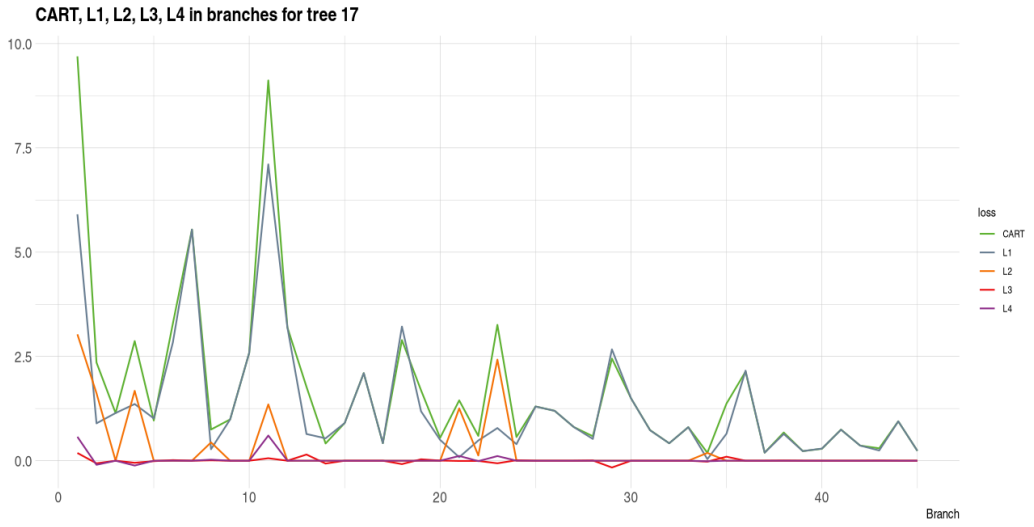

Figure 2.24: L_n, $L_{1,n}$, $L_{2,n}$, $L_{3,n}$ and $L_{4,n}$ in the cells of a tree in a random forest where an MCAR mechanism was present in the data set.

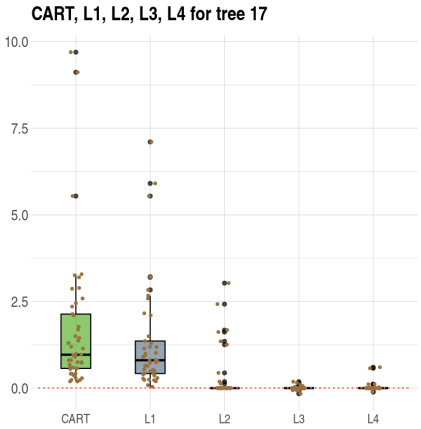

Figure 2.25: L_n, $L_{1,n}$, $L_{2,n}$, $L_{3,n}$ and $L_{4,n}$ boxplots for a tree in a random forest where an MCAR mechanism was present in the data set.

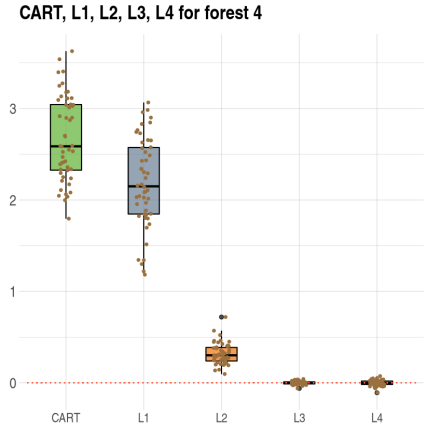

Figure 2.26: L_n, $L_{1,n}$, $L_{2,n}$, $L_{3,n}$ and $L_{4,n}$ boxplots for a random forest an MCAR mechanism was present in the data set, each point represents the mean value of a tree.

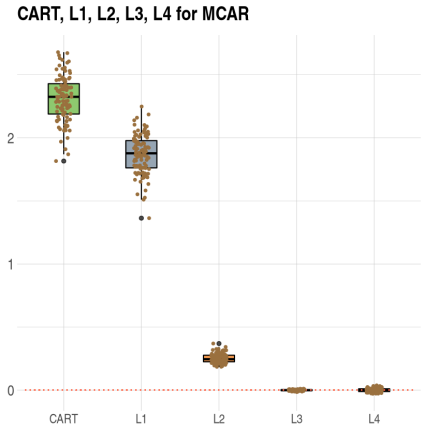

Figure 2.27: L_n, $L_{1,n}$, $L_{2,n}$, $L_{3,n}$ and $L_{4,n}$ boxplots when an MCAR mechanism was present in the data set, each point represents the mean value of a random forest.

2.6 Discussion and Future Work

In this chapter we developed a simulation study comparing the approach proposed in this work with three simple baselines corresponding to the deletion of observations with missing values, deletion of columns with missing values and single median imputation. Three imputation algorithms based on random forests were also consider and Missing Incorporated in Attributes (MIA) as state-of-the-art procedures which have been con-

sider in other simulation studies (Ishioka, 2013; Stekhoven and Bühlmann, 2011; Twala, Jones, and Hand, 2008; Josse et al., 2019). The algorithm proposed in this work (like MIA) handle missing values directly in the loss function consider in the construction of the trees. Two of the random forest-based imputation algorithms rely on the computation of extra structures like the so-called proximity matrix and improve the imputation iteratively. The remaining imputation algorithm corresponds to missForest where the imputation values of a feature is uploaded using the current imputation of all the other variables, treating the imputation of the missing values as a regression problem by itself, which is a similar strategy for other imputation algorithms like multiple imputation with chained equations (MICE).

With no surprise, listwise elimination and deletion of variables with missing values were the approaches with the worst performance, these methods should be avoided unless the percentage of observations or columns, respectively, with missing values is so low that they can be deleted without a severe harmful. For the rest of the algorithms considered in this simulation there is no clear advantage on either imputation-based methods or approaches that handle missing values directly in the loss function use in the construction of the trees. All these algorithms seem to be robust to the missing value generating process, which has also been pointed out as a property of *knn*-imputation and surrogate splits (Rieger, Hothorn, and Strobl, 2010). Although, the imputation based algorithms seem to create estimation of the regression function that are (aprox.) unbiased for all the scenarios, while the estimation obtained through MIA might suffer of severe bias.

In the study of the method proposed in this work, we also vary the percentage of missingness between 5% and 95% in one of the variables, observing that even when the percentage of missing values is high the algorithm is able to estimate the regression function with little deterioration in terms of the MSE. An important observation is that the number of possible assignations for a given cut (z, h) increases extremely rapid with the number of missing values in that variable since there are $2^{n_{miss}^{(h)}}$ possible assignations, which turns the algorithm to be prohibited in real applications. A possible solution for this severe lack is to select at random just a small percentage of all these assignations and select the assignation that maximizes the loss function between those selected at random. On the other hand, note that MIA only consider only three different assignations for the missing values allowing its computation even when there are many observations with missing values. It is important to observe that a simple baseline like median imputation shows similar performance to more complicated imputation algorithms or exhaustive computing methods, which make us wonder if all the extra effort that it is required is worthy for such a the little improvement.

Note that the mechanisms of missingness as well as the percentage of missing values considered are not necessarily comparable to real life data. Therefore a further work would include real life data sets that already contain missing values as other studies suggests (Hapfelmeier, Hothorn, and Ulm, 2012). A future study could also consider note only random forests based on the CART criterion but also conditional inference random forests (Hothorn, Hornik, and Zeileis, 2006; Rieger, Hothorn, and Strobl, 2010; Josse et al., 2019). Moreover, note that some algorithms to handle missing values like surrogate splits were not consider in this study, or even methods that are not based on recursive partitioning like *knn*-imputation (Troyanskaya et al., 2001), MICE (Van Buuren et al., 2006) or neural networks-based approaches like autoencoders (Chapter 4 is devoted to the study of the latter).

Finally, in Section 2.5 we decomposed the CART criterion with missing values as the sum of 4 new functions, giving some intuition on the meaning of these functions

and studied their behavior for the MCAR mechanism, however further work is needed in this direction, moving beyond the MCAR scenario and considering other mechanisms of missingness.

3

Consistency of a Random Forest Algorithm with Interval Imputation of Missing Entries for an Additive Model

Despite their use in practice, little is known about the mathematical mechanisms that drive random forests which has lead to a gap between theory and practice. Part of this gap can be explained by the bagging mechanism and the splitting criterion. Each of these processes introduces a source of randomness into the construction of the trees which makes the random forest algorithm very challenging to study in its full generality. One way to overcome the theoretical justifications of random forests is through simplified versions of the original procedure. This is often done by simply ignoring the bagging step and/or by replacing the splitting criterion with a more elementary splitting protocol.

Zhao (2000) and Cutler and Zhao (2001) introduce the Perfect Ensemble Random Trees (PERT) which became one of the pioneer works to study the consistency of ensembles of randomized trees. The criterion use in the PERT algorithm randomly chooses both the feature to split and the position of the cut. The authors conjecture that PERT is weakly consistent since all classifiers are almost uncorrelated. Centered forests is a simplified variant of the random forests algorithm, which assumes (i) no bootstrap subsampling, (ii) at each node of the trees a feature is uniformly chosen, and (iii) the split is selected at the midpoint of the values of the selected variable. The centered forest rule was first formally analyzed by Breiman (2004), and later by Biau, Devroye, and Lugosi (2008) and Scornet (2016), who proved that the method is consistent (both for classification and regression) provided $p \to \infty$ and $n/2^p \to \infty$ (the average number of points at the final nodes). It was pointed out by Lin and Jeon (2006) that Layered Nearest Neighbors (LNN), for which consistency conditions are met, are intimately connected to this algorithm. In Purely Random Forests (PURF) at each step, one of the current nodes and one variable are chosen uniformly at random. The selected node is then split along the selected variable at a random location inside the limits of the cell. Biau, Devroye, and Lugosi (2008) study the consistency of PURF and Genuer (2012) proves that PURF satisfies a minimax rate over Lipschitz functions. Note that trees constructed in PURF are built in a data-independent manner, that is without looking at the training sample $\mathcal{D}_n$. Biau (2012) establishes the consistency of a variant in which at each step all the current final nodes are independently split at the midpoint of a variable selected at random. Scornet, Biau, and Vert (2015) establishes the first consistency result for the original random forest algorithm assuming a regression additive models. Asymptotic normality for Breiman's infinite forests were proven by Wager and Athey (2018) simplifying the splitting step and Mentch and Hooker (2016) proved a similar result for finite forests. On domain-specific adaptations of random

forests some results on the consistency includes quantile regression (Meinshausen, 2006), survival analysis (Ishwaran and Kogalur, 2010), online forests (Denil, Matheson, and Freitas, 2013) and Reinforcement Learning Trees (Zhu, Zeng, and Kosorok, 2015).

3.1 Consistency of a Random Forest Algorithm with Missing Entries for an Additive Model

In this section we give a consistency result in the context of an additive regression in presence of missing data accordingly to a missing completely at random (MCAR) paradigm. Additive regression models, which decompose the regression function as a sum of univariate functions, are flexible and easy to interpret, providing a good trade-off between model complexity, calculation time and interpretation. We have taken the previous work of Scornet, Biau, and Vert (2015) as inspiration for this part, however the proof of the results differs from theirs in several parts. In particular, we do not need to make comparisons between trees in the whole structure, but just need to study the behavior of the final cells at a price of a strong assumption on the number of points belonging to these nodes. We consider an additive regression model satisfying the following properties.

Hypothesis 1. *The response variable Y has the form*

$$Y = \sum_{j=1}^{p} m_j(\mathbf{X}^{(j)}) + \varepsilon$$

where $\mathbf{X}$ is uniformly distributed over $[0,1]^p$, ε is an independent Gaussian centered noise with finite variance $\sigma^2 > 0$ and each component m_j is continuous.

The fact that Y has an additive structure over the predictor variables is important for the statement of Technical Lemma 1. For instance, we show in Example 1 a case in which m does not have this structure and the result does not hold anymore. On the other hand, the proof of Technical Lemma 3 lies on the Gaussian structure of the errors.

Hypothesis 2. *The random variables $\mathbf{X}_i^{(h)}$ are not observed (missing) by following an MCAR mechanism. The probability of missingness $p_n^{(h)} = \mathbb{P}\left[\mathbf{M}^{(h)} = 1\right]$ only depends on the size n of the sample $\mathcal{D}_n$ and $\lim_{n\to\infty} p_n^{(h)} = c^{(h)}$ where $0 < c^{(h)} < 1$ is constant for all $h \in \{1, \dots, p\}$.*

Ideally an assumption allowing $p_n^{(h)} \to 1$ is desirable. However, the final statement for the proof of Theorem 1 would not hold as it is, since we were not able to show that the mean of the observations assigned to the left and the mean of those assigned to the right converges to the same value. In general, we have used extensively the MCAR assumption to prove our results.

Theorem 1

Assume that Hypothesis 1 and 2 hold. Denote by q_n the minimum number of points in each final cell of the random trees. Then, under the condition $q_n \to \infty$, the random forest estimator with missing values is universally consistent in probability, i.e., for all $\xi, \rho > 0$ there exists $N \in \mathbb{N}^\star$, such that for all $n > N$

$$\mathbb{P}\Big[|m_n(\mathbf{X}) - m(\mathbf{X})| \leq \xi\Big] \geq 1 - \rho$$

The fact that $q_n \to \infty$ implies that the number of points selected in each tree, a_n tends to infinity too. Hence, in the sequel, we assume that $a_n \to \infty$ and omit the dependence of the trees over a_n. Moreover, the condition that $q_n \to \infty$ is sub-optimal, and we think that with some extra effort the results presented in this work can be adapted to a more classical (and more powerful) condition like $a_n/t_n \to \infty$. In this case, it might be possible that not every tree would be consistent but the random forest would converge to the regression function.

We define, for any subset $A \subset \mathcal{X}$, the variation of m within A as

$$\Delta(m, A) = \sup_{\mathbf{x}, \mathbf{x}' \in A} |m(\mathbf{x}) - m(\mathbf{x}')|$$

Furthermore, we denote by $A_{s(n)}(\mathbf{X}, \Theta)$ the final cell of the tree built with the random variable Θ that contains $\mathbf{X}$, where $s(n)$ is the number of cuts necessary to construct the cell in the tree. After the last step of the tree construction, we end with a collection of imputed values that corresponds to the last imputed sample $(\widehat{\mathbf{X}}_{1,out}, \dots, \widehat{\mathbf{X}}_{n,out})$. Each of these vectors are non ambiguously assigned to a specific final cell of the tree. In all that follows, when we write an imputation $\widehat{\mathbf{X}}$ without any specification of *in* or *out*, we refer to the final imputation.

The proof of Theorem 1 relies on Proposition 1 below, which states that the variation of the regression function within a final cell tree is small for n large enough and allows to control the error of our predictor. In the sequel, we will abuse of the notation and will not write the dependence of the cells over $\mathbf{X}$ and Θ.

Proposition 1

Assume that Hypothesis 1 and 2 hold. Then,

$$\Delta(m, A_{s(n)}) \to 0, \quad \text{almost surely.}$$

For all $\mathbf{x} \in [0,1]^p$, we denote by $L^\star\left(A_{s(n)}, d, w\right)$ the theoretical CART criterion over the cell $A_{s(n)}$ evaluated at a cut $d \in \mathcal{C}_{A_{s(n)}}$ and assignation $w \in \mathcal{W}$. Let $(d^\star_{s(n)}, w^\star_{s(n)})$ be the optimal couple (cut, assignation) of the cell $A_{s(n)}$ for the theoretical criterion and let $(\widehat{d}_{s(n)}, \widehat{w}_{s(n)})$ be the optimal couple (cut, assignation) of the cell $A_{s(n)}$ for the empirical criterion.

For ease of understanding we have divided the proof in the next sections. In Section 3.2 we enunciate and prove Lemma 1 which establishes that if the theoretical CART criterion, evaluated in what would be the best cut of the final nodes, tends to zero in probability then the variation of the regression function tends to zero as well. In Section 3.3 we enunciate and prove Lemma 2 which establishes that the empirical CART criterion, evaluated in what would be the best empirical cut of the final nodes converges to zero in probability. Finally in Section 3.4 we prove Proposition 1 as a consequence of Lemmas 1 and 2. To prove Theorem 1 we make use of another regression estimate m''_n which is shown to be consistent by Proposition 1 and we show that our estimator and m''_n are asymptotically equivalent which completes the proof of Theorem 1.

3.2 Low Values of the Theoretical CART Criterion Implies Minimum Variation of the Regression Function

A key part for the statement of Theorem 1 is to show that the regression function tends to zero on the final cells in the trees built with our approach. In this section we enunciate Lemma 1 which shows that if the theoretical CART criterion in the final cells, evaluated at the best theoretical cut, tends to zero in probability then the variation of the regression function in final cells tends to zero. The proof of Lemma 1 relies on Technical Lemmas 1 and 2.

Lemma 1

Assume that Hypothesis 1 and 2 are satisfied and fix $\mathbf{x} \in [0,1]^p$. Then for all $\rho, \xi > 0$, there exists $N \in \mathbb{N}^\star$ such that, for all $n \geq N$ if $\mathbb{P}\left[L^\star\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) \leq \xi\right] \geq 1-\rho$, then

$$\Delta(m, A_{s(n)}(\mathbf{x})) \to 0, \quad \text{almost surely.}$$

For the proof of Lemma 1, we first need a technical lemma which states that if $L^\star = 0$ for all cuts in a cell, using as a rule of assignment the mechanism of missingness, then the regression function is constant on that particular cell.

Technical Lemma 1

Assume that Hypothesis 1 and 2 are satisfied and for a cell A, $L^\star(A, d, \tilde{w}_{A,d}) \equiv 0$ for all cuts $d = (h, z) \in \mathcal{C}_A$, where

$$\tilde{w}_{A,d} = \mathbb{P}\left[a^{(h)} \leq \mathbf{X}^{(h)} < z | \mathbf{X} \in A, \mathbf{M}^{(h)} = 1\right].$$

Then, m is constant on the cell A.

Proof of Technical Lemma 1 Without loss of generality, we will assume that the cut $d = (1, z)$ is performed in the first direction so that $h = 1$ and that the bounds of the cell A are a on the left and b on the right. We omit the direction h and simply note X instead of $\mathbf{X}^{(1)}$. Note that if $\widehat{\mathbf{X}}_{out}$ is assigned to the left child node using $\tilde{w}_{A,d}$, which does not depend on Y, then $\widehat{\mathbf{X}}_{out}$ follows the same distribution than $\mathbf{X}$ and then our theoretical CART criterion is similar to the usual one for every cut $d = (h, z) \in \mathcal{C}_A$, that is,

$$L^\star\left(d, \tilde{w}_{A,d}\right) = \mathbb{P}\left[a \leq X < z | \mathbf{X} \in A\right] \mathbb{P}\left[z \leq X \leq b | \mathbf{X} \in A\right] \left(\mu_{A_L} - \mu_{A_R}\right)^2$$

where the notation μ_{A_L} (resp. μ_{A_R}) holds for the conditional expected value of Y given $\widehat{\mathbf{X}}_{in} \in A$ and $a \leq X < z$ (resp. $z \leq X \leq b$). Because $\mathbf{X}$ is uniformly distributed over $[0,1]^p$,

$$\mathbb{P}\left[a \leq X < z | \mathbf{X} \in A\right] = \frac{z-a}{b-a}$$

and

$$\mathbb{P}\left[z \leq X \leq b | \mathbf{X} \in A\right] = \frac{b-z}{b-a}$$

Next, we need to understand the distribution of Y conditionally to $\widehat{\mathbf{X}}_{in} \in A$. This random variable is a mixture of the values of Y such that $\mathbf{X} \in A$ and the ones that where assigned to the cell A through the notion of the variable $\widehat{\mathbf{X}}_{in}$. Since the vector $(\widehat{\mathbf{X}}_{in}, Y)$ has a density, we can give a precise meaning to the quantity

$$\tilde{m}(x) = \mathbb{E}\left[Y | \widehat{\mathbf{X}}_{in} = \mathbf{x}\right], \tag{3.1}$$

for all $\mathbf{x} \in A$. By Hypothesis 1, we introduce the notation $Y^{(j)} = m_j(X^{(j)}) + \epsilon^{(j)}$, where $\epsilon^{(j)} \sim \mathcal{N}(0, \sigma^2/p)$ so that we have $Y = \sum_{j=1}^{p} Y^{(j)}$. Since under the condition $\mathbf{X} \in A$, the random variable $\mathbf{X}$ is also uniformly distributed on the cell A, we have that

$$\begin{aligned}
\mu_{A_L} &= \mathbb{E}\left[\sum_{j=1}^{p} Y^{(j)} | a \leq X < z, \widehat{\mathbf{X}}_{in} \in A\right] \\
&= \sum_{j \geq 2} \mathbb{E}\left[Y^{(j)} | \widehat{\mathbf{X}}_{in} \in A\right] + \mathbb{E}\left[Y^{(1)} | a \leq X < z, \widehat{\mathbf{X}}_{in} \in A\right] \\
&= \tilde{K} + \frac{1}{z-a} C_a^z
\end{aligned}$$

where $\tilde{K}$ does not depend on z, $C_x^y = \int_x^y \tilde{m}_1(t)dt$ and $\tilde{m}_1(t) = \mathbb{E}\left[Y^{(1)} | \widehat{\mathbf{X}}_{in} \in A,\ \mathbf{X}^{(1)} = t\right]$. Analogously, we have that

$$\mu_{A_R} = \tilde{K} - \frac{1}{b-z} C_a^z + \frac{1}{b-z} C_a^b$$

Therefore,

$$L^\star\left(A, d, \tilde{w}_{A,d}\right) = \left(\frac{z-a}{b-a}\right)\left(\frac{b-z}{b-a}\right)\left(\frac{1}{z-a} C_a^z + \frac{1}{b-z} C_a^z - \frac{1}{b-z} C_a^b\right)^2$$

$$= \frac{1}{(z-a)(b-z)}\left(C_a^z - \frac{z-a}{b-a} C_a^b\right)^2$$

Since $L^\star\left(h, z, \tilde{w}_{A,d}\right) = 0$ by assumption, we obtain that for any $a \leq z \leq b$,

$$C_a^z = \frac{z-a}{b-a} C_a^b.$$

This proves that $z \mapsto C_a^z$ is linear in z and thus, $\tilde{m}_1$ is constant on $[a, b]$. Since the law of $Y^{(1)}$ is a mixture of distribution such that $M^{(1)} = 0$ and $M^{(1)} = 1$, we have that

$$\tilde{m}_1(t) = (1 - p^{(1)}) m_1(t) + p^{(1)} \mathbb{E}\left[Y^{(1)} | \widehat{\mathbf{X}}_{in} \in A,\ \mathbf{X}^{(1)} = t, \mathbf{M}^{(1)} = 1\right]$$

where the second term does not depend on t (since the value is missing). This forces the function m_1 to be constant on the interval $[a, b]$. And, by additivity, the function m is constant on A.

□

Example 1

If m is non additive then we could have that the CART criterion equals zero even when m is not constant. For example, consider the function $m(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \mathbf{x}^{(1)} + \mathbf{x}^{(2)} - 2\mathbf{x}^{(1)}\mathbf{x}^{(2)}$, the cell $A = [0, 1]^p$ and a cut $d = (1, z)$ $(z \in (0, 1))$, then

$$\begin{aligned}\mu_{A_L} &= \frac{1}{z}\int_0^z \int_0^1 (\mathbf{x}^{(1)} + \mathbf{x}^{(2)} - 2\mathbf{x}^{(1)}\mathbf{x}^{(2)}) d\mathbf{x}^{(2)} d\mathbf{x}^{(1)} \\ &= \frac{1}{z}\int_0^z (\mathbf{x}^{(1)} + \frac{1}{2} - \mathbf{x}^{(1)}) d\mathbf{x}^{(1)} \\ &= \frac{1}{2}\end{aligned}$$

Note that μ_{A_L} does not depend on z, similarly we can show that $\mu_{A_R} = 1/2$ and therefore $L^\star = 0$ for all cuts even when m is not constant.

We need a second technical lemma which states that any sub-cell within a cell for which each split gives a small value of $L^\star$ has also a uniformly small value of the CART criterion $L^\star$.

Technical Lemma 2

Assume constructed the imputation $\widehat{\mathbf{X}}_{in}$. Let $\epsilon > 0$, then there exists $\delta > 0$ such that for all cell A and cut $d \in \mathcal{C}_A$ satisfying $L^\star(A, d, \tilde{w}_{A,d}) \leq \delta$, we have

$$L^\star(B, d, \tilde{w}_{A,d}) \leq \epsilon$$

for all cell $B \subset A$, where both notions of CART criteria are constructed with the same vector of imputation $\widehat{\mathbf{X}}_{in}$.

Proof of Technical Lemma 2 We first prove the unidimensional case. Without loss of generality, we can assume that the cell A is the interval $[0, 1]$ and we first consider the cell B of the form $[0, a]$ with $0 \leq a \leq 1$. Following the lines of Technical Lemma 1, we have that

$$L^\star(A, z, \tilde{w}_{A,d}) = \frac{1}{z(1-z)}\left(C_0^z - C_0^1 z\right)^2$$

where for all x and y, $C_x^y = \int_x^y \tilde{m}(t)dt$ and $\tilde{m}$ is defined as in Equation (3.1). Evaluating the CART criterion at the value $z = a$, we have

$$\frac{1}{a(1-a)}\left(C_0^a - C_0^1 a\right)^2 \leq \delta.$$

For the cell B, at a cut level $0 < z < a$, we have that

$$\begin{aligned} L^\star(B, z, \tilde{w}_{A,d}) &= \frac{1}{z(a-z)} \left(C_0^z - C_0^a \frac{z}{a} \right)^2 \\ &= \frac{1}{z(a-z)} \left(C_0^z - C_0^1 z + C_0^1 z - C_0^a \frac{z}{a} \right)^2 \\ &\leq \frac{2}{z(a-z)} \left(C_0^z - C_0^1 z \right)^2 + \frac{2z}{a-z} \left(C_0^1 - \frac{C_0^a}{a} \right)^2 \\ &\leq 2 \frac{1-z}{a-z} L^\star(A, z, \tilde{w}_{A,d}) + \frac{2z(1-a)}{a(a-z)} \delta \\ &\leq 2 \frac{a(1-z) + z(1-a)}{a(a-z)} \delta \leq \frac{2}{a(a-z)} \delta. \end{aligned}$$

But since the function $z \mapsto C_0^z$ is differentiable, we have that $C_0^z = C_0^a - (a-z)\tilde{m}(a) + o(a-z)$ when $z \to a$ by a Taylor expansion. This shows that $L^\star(B, z, \tilde{w}_{A,d}) \to 0$ when $z \to a$ and then there exists a $\delta_0 > 0$ such that $L^\star(B, z, \tilde{w}_{A,d}) \leq \epsilon$ when $(a-z) \leq \delta_0$. Then, using the previous inequalities for $0 < z < a - \delta_0$, we have that for $\delta = \epsilon \delta_0 a/2$, $L^\star(B, z, \tilde{w}_{A,d}) \leq \epsilon$. In the same way, we generalize the previous arguments for $A = [0,1]^p$ and for a specific type of cell

$$B = \{ \mathbf{x} \in A : \mathbf{x}^{(1)} \leq a \}$$

with $0 < a < 1$. The values C_y^z are then replaced by $\int_y^z \int_0^1 \cdots \int_0^1 \tilde{m}(t) dt_1 dt_2 \ldots dt_p$. The result can be repeated for the case $B = \{ \mathbf{x} \in A : \mathbf{x}^{(1)} \geq a \}$.

For the general case of $B \subset A$ we see that any B can be obtained by a finite sequence of $B = B_k \subset B_{k-1} \subset \ldots B_1 \subset A$ of subset constructed by the scheme described above. This finishes the proof.

□

Proof of Lemma 1 We will show that $\Delta(m, A_{s(n)}(\mathbf{x})) \to 0$ a.s. by contradiction. We assume that with positive probability, there exists a positive constant $c > 0$ and a sub-sequence $\phi(n)$ of cells $A_{s(\phi(n))}$ such that $\Delta(m, A_{s(\phi(n))}(\mathbf{x})) > c$. This means that in each set $A_{s(\phi(n))}(\mathbf{x})$ one can find a pair of elements (x_n, y_n) such that

$$|m(x_n) - m(y_n)| > c.$$

The sequences $(x_n)_n$ and $(y_n)_n$ belong to the compact set $[0,1]^p$ so one can extract a sub-sequence $\psi(n)$ such that $(x_n)_n$ and $(y_n)_n$ converge respectively to two points x and y. By continuity of m, we have that

$$|m(x) - m(y)| > c.$$

At the cost of taking an x' and y' close to x and y, satisfying

$$|m(x') - m(y')| > \frac{c}{2},$$

and such that, for n large enough, all the cells $A_{s(\phi(n))}$ contain the pair of points x' and y'.

By hypothesis, we know that

$$\sup_{d \in \mathcal{C}_{A_{s(\phi \circ \psi(n))}}} L^\star(A_{s(\phi \circ \psi(n))}, d, \tilde{w}) \to 0 \quad \text{in probability}$$

where we just wrote $\tilde{w}$ for the choice of the assignation given in Technical Lemma 1. So one can extract a subsequence $\chi(n)$ such that the $\sup_d L^\star(A_{s(\phi\circ\psi\circ\chi(n))}, d, \tilde{w})$ converges to 0 almost surely. For simplicity, we keep denoting n for the sub-sequence $\phi\circ\psi\circ\chi(n)$ in the following part of the proof. Lastly, we define the sequence of cells $(C_i)_{i\geq 1}$ such that

$$C_i = \bigcap_{k=1}^{i} A_{s(k)}.$$

These cells form a non increasing sequence for the inclusion order and for each i, $C_i \subset A_{s(i)}$. The cells C_i inherit the same vector of imputation as for $A_{s(i)}$. We can use Technical Lemma 2 to get that $\sup_{d\in\mathcal{C}_{A_{s(n)}}} L^\star(C_n, d, \tilde{w}) \to 0$ a.s. Since the sequence of cells $(C_i)_i$ is a non increasing sequence, there exists a cell, denoted C_∞ that is the limit of the cells C_i when $i \to \infty$ in the sense

$$C_\infty = \bigcap_{i\geq 1} C_i.$$

To see that, one can write $C_i = \prod_h [a_i^{(h)}, b_i^{(h)}]$ and take the limits of the sequences $(a_i^{(h)})_i$ and $(b_i^{(h)})_i$. The objective is to show that there is no other possibility than having $L^\star(C_\infty, d, \tilde{w}) = 0$ for every cut d. For a cell A, the CART criterion $L^\star(A, d, \tilde{w}_{A,d})$ has a continuous behavior with respect to A. Indeed, we see that

$$L^\star\left(A, d, \tilde{w}_{A,d}\right) = \mathbb{P}\left[a^{(h)} \leq \mathbf{X}^{(h)} < z | \mathbf{X} \in A\right] \mathbb{P}\left[z \leq \mathbf{X}^{(h)} \leq b^{(h)} | \mathbf{X} \in A\right] \left(\widehat{\mu}_{A_L} - \widehat{\mu}_{A_R}\right)^2$$

which is a product of three terms that are uniformly continuous in A (since m is a continuous function) with respect to the distance between cells given by $\Delta(A, B) = \max_h |x_A^{(h)} - x_B^{(h)}| + \max_h |y_A^{(h)} - y_B^{(h)}|$ where $x_A^{(h)}$ and $y_A^{(h)}$ are defined as $A = \bigcap_h [x_A^{(h)}, y_A^{(h)}]$. Since $C_i \to C_\infty$ for the distance Δ, we have that for all $\epsilon > 0$, for all i large enough, for all cut d of the final cell C_∞,

$$\left|L^\star\left(C_i, d, \tilde{w}\right) - L^\star\left(C_\infty, d, \tilde{w}\right)\right| \leq \epsilon.$$

But since the $L^\star\left(C_i, d, \tilde{w}\right)$ converges almost surely uniformly in d to 0, and ϵ is arbitrary, for every cut d inside the valid cuts of C_∞, we have that $L^\star(C_\infty, d, \tilde{w}) = 0$. But then by Technical Lemma 1, the function m has to be constant in the cell C_∞. But the points x' and y' do belong to the cell C_∞ since they belong to each of the cells in the intersection. So $m(x') = m(y')$ which contradicts the fact that $|m(x') - m(y')| > c/2$. This proves that

$$\Delta(m, A_{s(n)}(\mathbf{x})) \to 0 \quad \text{almost surely.}$$

□

3.3 The Empirical CART Criterion Converges to Zero in Probability

Since our trees are built using the empirical CART criterion, it is necessary to study its asymptotic behavior. In particular Lemma 2 establishes that asymptotically the empirical CART criterion in the final cells, evaluated at the best empirical cut tends to zero in probability, which is a desirable property and will be used in thee proof of Proposition 1 and Theorem 1. To prove Lemma 2, we are going to make use of Technical Lemma 3 below which establishes that for cells nearby a final cell of the tree the empirical CART criterion cannot change too much.

Lemma 2

Assume that Hypothesis 1 and 2 are satisfied and fix $\mathbf{x} \in [0,1]^p$. Then for all $\rho, \xi > 0$, there exists $N \in \mathbb{N}^\star$ such that, for all $n \geq N$

$$\mathbb{P}\left[L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \leq \xi\right] \geq 1 - \rho$$

Remember that $A_{s(n)}$ denotes the final cell of the tree built with the random variable Θ that contains $\mathbf{X}$, where $s(n)$ is the number of cuts necessary to construct the cell. Similarly, A_k is the same cell but where only the first k cuts have been performed.

Technical Lemma 3

Assume that Hypothesis 1 and 2 hold and fix $\mathbf{x} \in [0,1]^p$. For all $\rho, \xi > 0$ there exists $N \in \mathbb{N}^\star$ such that for all $n \geq N$ there exists $k_0(n) \in \mathbb{N}^\star$ such that

$$\mathbb{P}\left[\left|L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) - L_n\left(A_k, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right)\right| \leq \xi\right] \geq 1 - \rho$$

for all $k \geq k_0(n)$

Proof of Technical Lemma 3 Fix $\alpha, \rho > 0$ and consider the following standard inequality on a Gaussian tail

$$\mathbb{P}[\varepsilon_1 \geq \alpha] \leq \frac{\sigma}{\alpha\sqrt{2\pi}} \exp\left(-\frac{\alpha^2}{2\sigma^2}\right)$$

then, simple calculations show that, for all $n \in \mathbb{N}^\star$

$$\mathbb{P}\left\{\left|\sum_{i=1}^{n} \varepsilon_i\right| \geq n\alpha\right\} \leq \frac{\sigma}{\alpha\sqrt{n}} \exp\left\{-\frac{\alpha^2 n}{2\sigma^2}\right\} \tag{3.2}$$

Note that there are at most $n(n+1)/2$ sets of the form $\{i : \mathbf{X}_i^{(h)} \in [a_n, b_n], \mathbf{M}_i^{(h)} = 0\}$ for $0 \leq a_n < b_n \leq 1$. On the other hand, let $(Y_{(1)}, \ldots, Y_{(n)})$ be the order vector of Y, since missing observations are assigned to the cell using Y and maximizing the CART criterion, this implies that close values of Y must be assigned to the same cell, thus once again note that there are at most $n(n+1)/2$ sets of the form $\{i : Y_{(i)} \in [a_n, b_n], \mathbf{M}_i^{(h)} = 1\}$ for $0 \leq a_n < b_n \leq 1$.

We deduce from Equation (3.2) and the union bound, that there exists $N_1 \in \mathbb{N}^\star$ such that, with probability at least $1 - \rho$, for all $n \geq N_1$ and all $0 \leq a_n < b_n \leq 1$ satisfying $\widehat{N}\left(\prod_{h=1}^{p}[a_n^{(h)}, b_n^{(h)}]\right) \geq q_n$,

$$\left|\frac{1}{\widehat{N}\left(\prod_{h=1}^{p}[a_n^{(h)}, b_n^{(h)}]\right)} \sum_{i=1}^{n} \varepsilon_i \mathbb{1}_{\widehat{\mathbf{X}}_i \in A}\right| \leq \frac{\sigma n^{4p}}{\alpha\sqrt{n}} \exp\left\{-\frac{\alpha^2 n}{2\sigma^2}\right\} \leq \alpha \tag{3.3}$$

Furthermore, making use of the same ideas and applying the inequality $\mathbb{P}[\chi^2(n) \geq 5n] \leq e^{-n}$ (for interested readers, see Laurent and Massart (2000)), there exists $N_2 \in \mathbb{N}^\star$ such

that, with probability at least $1-\rho$ for all $n \geq N_2$ and all $0 \leq a_n < b_n \leq 1$ satisfying $\widehat{N}\left(\prod_{h=1}^{p}[a_n^{(h)}, b_n^{(h)}]\right) \geq q_n$,

$$\frac{1}{\widehat{N}\left(\prod_{h=1}^{p}[a_n^{(h)}, b_n^{(h)}]\right)} \sum_{i=1}^{n} \varepsilon_i^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A} \leq \tilde{\sigma}^2 \tag{3.4}$$

where $\tilde{\sigma}^2$ is a positive constant, depending only on ρ.

Since $(A_k)_k$ is a decreasing sequence of compact sets, for every $\xi > 0$ there exists k_0 such that, for all $k \geq k_0$

$$\max\left(\|\mathbf{a}_k - \mathbf{a}_{s(n)}\|_\infty, \|\mathbf{b}_k - \mathbf{b}_{s(n)}\|_\infty\right) \leq \xi \tag{3.5}$$

where $\mathbf{a}_k = \left(a_k^{(1)}, \ldots, a_k^{(p)}\right) \in [0,1]^p$ and $\mathbf{b}_k = \left(b_k^{(1)}, \ldots, b_k^{(p)}\right) \in [0,1]^p$ such that

$$A_k = \prod_{j=1}^{p}\left[a_k^{(j)}, b_k^{(j)}\right]$$

We define $\mathbf{a}_{s(n)}$ and $\mathbf{b}_{s(n)}$ analogously for the cell $A_{s(n)}$.

Because $\mathbf{X}$ is uniformly distributed in the hypercube $[0,1]^p$ and the missing process follows an MCAR mechanism we have that, for any cell A, $\widehat{N}_{obs}(A)/\widehat{N}_{obs}([0,1]^p) \to \mathrm{Vol}(A)$ almost surely as $n \to \infty$. Furthermore, for any $k \geq k_0(n)$, Equation (3.5) implies that $\mathrm{Vol}(A_k) \leq (1+2\xi)^p \mathrm{Vol}(A_{s(n)})$. Then there exists $N_3 \in \mathbb{N}^\star$ such that for all $n \geq N_3$, $\widehat{N}_{obs}(A_k) \leq (1+3\xi)^p \widehat{N}_{obs}(A_{s(n)})$, and

$$\widehat{N}_{obs}(A_k \setminus A_{s(n)}) \leq \xi' \widehat{N}_{obs}(A_{s(n)}) \leq \xi' \widehat{N}_{obs}(A_k) \tag{3.6}$$

where $\xi' = (1+3\xi)^p - 1$. On the other hand, for any fixed n, the quantity $u_k = \widehat{N}_{miss}(A_k) - \widehat{N}_{miss}(A_{s(n)})$ converges to 0 almost surely as $k \to \infty$. Now taking $n \geq N_3$ fixed we have that there exists $k_1(n)$ such that for all $k \geq k_1(n)$, with probability at least $1-\rho$, we have

$$\begin{aligned}
\widehat{N}(A_k \backslash A_{s(n)}) &= (\widehat{N}_{obs}(A_k) + \widehat{N}_{miss}(A_k)) - (\widehat{N}_{obs}(A_{s(n)}) + \widehat{N}_{miss}(A_{s(n)})) \\
&= \widehat{N}_{obs}(A_k \setminus A_{s(n)}) + u_k \\
&\leq 2\xi' \widehat{N}_{obs}(A_{s(n)})
\end{aligned} \tag{3.7}$$

where we have used Equation (3.6) and the fact that convergence almost sure implies convergence in probability. For the rest of the proof, we take $k \geq \max\{k_0(n), k_1(n)\}$ and assume that Equations (3.3), (3.4) and (3.7) are satisfied, which occurs with probability at least $1-3\rho$ for every $n > N$ with $N = \max\{N_1, N_2, N_3\}$.

Note that $A_k \setminus A_{s(n)}$ is either a final node, contains at least one final node or is empty (in which case the result holds trivially). Since each final node contains at least q_n points then for q_n sufficiently large, using Equation (3.3) we conclude that $|\widehat{Y}_{A_k}| \leq \|m\|_\infty + \alpha$, $|\widehat{Y}_{A_{s(n)}}| \leq \|m\|_\infty + \alpha$ and $|\widehat{Y}_{A_k \backslash A_{s(n)}}| \leq \|m\|_\infty + \alpha$. We now use the following decomposition

$$\left|L_n\left(A_k, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) - L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right)\right| \leq K_0 + K_L + K_R$$

where the three terms K_0, K_L and K_R are given by

$$K_0 = \left|\frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_k})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k} - \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{s(n)}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}}\right|,$$

$$K_L = \Big| \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{L,k}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k, \widehat{\mathbf{X}}_i^{(\widehat{h})} < \widehat{z}} - \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{L,s(n)}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}, \widehat{\mathbf{X}}_i^{(\widehat{h})} < \widehat{z}} \Big|,$$

$$K_R = \Big| \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{R,k}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k, \widehat{\mathbf{X}}_i^{(\widehat{h})} \geq \widehat{z}} - \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{R,s(n)}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}, \widehat{\mathbf{X}}_i^{(\widehat{h})} \geq \widehat{z}} \Big|.$$

We first consider the term K_0 that can be upper bounded once again by using a similar split in $K_0 \leq K_{0,1} + K_{0,2} + K_{0,3}$ where

$$K_{0,1} = \Big| \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_k})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} - \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{s(n)}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} \Big|$$

$$K_{0,2} = \Big| \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{s(n)}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} - \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{s(n)}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} \Big|$$

$$K_{0,3} = \Big| \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_k})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k \setminus A_{s(n)}} \Big|$$

For $K_{0,1}$, observe that

$$\begin{aligned} |\widehat{Y}_{A_k} - \widehat{Y}_{A_{s(n)}}| &= \Big| \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} Y_i \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k \setminus A_{s(n)}} + \frac{1}{\widehat{N}(A_k)} \sum_{i=1}^{n} Y_i \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} - \widehat{Y}_{A_{s(n)}} \Big| \\ &\leq \frac{\widehat{N}(A_k \setminus A_{s(n)})}{\widehat{N}(A_k)} |\widehat{Y}_{A_k \setminus A_{s(n)}} - \widehat{Y}_{A_{s(n)}}| \\ &\leq 4\xi'(\|m\|_\infty + \alpha) \end{aligned}$$

Hence,

$$\begin{aligned} K_{0,1} &\leq \frac{2}{\widehat{N}(A_k)} |\widehat{Y}_{A_{s(n)}} - \widehat{Y}_{A_k}| \left| \sum_{i=1}^{n} \left(Y_i + \frac{\widehat{Y}_{A_{s(n)}} + \widehat{Y}_{A_k}}{2} \right) \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} \right| \\ &\leq \frac{8\xi'(\|m\|_\infty + \alpha)}{\widehat{N}(A_k)} \left[\left| \widehat{N}(A_{s(n)}) \widehat{Y}_{A_{s(n)}} \right| + \left| \frac{\widehat{Y}_{A_{s(n)}} + \widehat{Y}_{A_k}}{2} \widehat{N}(A_{s(n)}) \right| \right] \\ &\leq 16\xi'(\|m\|_\infty + \alpha)^2 \end{aligned}$$

For the term $K_{0,2}$, with the help of Equation (3.4) observe that

$$\begin{aligned}
K_{0,2} &\leq \frac{\widehat{N}(A_k \setminus A_{s(n)})}{\widehat{N}(A_k)} \Big| \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} (Y_i - \widehat{Y}_{A_{s(n)}})^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} \Big| \\
&\leq 2\xi' \Big| \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} Y_i^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} + \widehat{Y}_{A_{s(n)}}^2 \Big| \\
&\leq 2\xi' \Bigg[(\|m\|_\infty + \alpha)^2 + \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} m^2(\mathbf{X}_i) \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} \\
&\qquad + \Big| \frac{2}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} m(\mathbf{X}_i) \varepsilon_i \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} \Big| + \frac{1}{\widehat{N}(A_{s(n)})} \sum_{i=1}^{n} \varepsilon_i^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}} \Bigg] \\
&\leq 2\xi' \left[(\|m\|_\infty + \alpha)^2 + \|m\|_\infty^2 + 2\|m\|_\infty \alpha + \tilde{\sigma}^2 \right]
\end{aligned}$$

Regarding $K_{0,3}$, observe that

$$\begin{aligned}
K_{0,3} &\leq 2\xi' \Big| \frac{1}{\widehat{N}(A_k \setminus A_{s(n)})} \sum_{i=1}^{n} (Y_i^2 + 2Y_i \widehat{Y}_{A_k} + \widehat{Y}_{A_k}^2) \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k \setminus A_{s(n)}} \Big| \\
&\leq 2\xi' \Bigg[\Big| \frac{1}{\widehat{N}(A_k \setminus A_{s(n)})} \sum_{i=1}^{n} (m(\mathbf{X}_i) + \varepsilon_i)^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k \setminus A_{s(n)}} \Big| \\
&\qquad + \frac{2}{\widehat{N}(A_k \setminus A_{s(n)})} (\|m\|_\infty + \alpha) \Big| \sum_{i=1}^{n} (m(\mathbf{X}_i) + \varepsilon_i) \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k \setminus A_{s(n)}} \Big| \\
&\qquad + (\|m\|_\infty + \alpha)^2 \Bigg] \\
&\leq 2\xi' \Bigg[\|m\|_\infty^2 + 2\|m\|_\infty \Big| \frac{1}{\widehat{N}(A_k \setminus A_{s(n)})} \sum_{i=1}^{n} \varepsilon_i \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k \setminus A_{s(n)}} \Big| \\
&\qquad + \frac{1}{\widehat{N}(A_k \setminus A_{s(n)})} \sum_{i=1}^{n} \varepsilon_i^2 \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_k \setminus A_{s(n)}} + 2(\|m\|_\infty + \alpha)^2 + (\|m\|_\infty + \alpha)^2 \Bigg] \\
&\leq 2\xi' [3(\|m\|_\infty + \alpha)^2 + \|m\|_\infty^2 + 2\|m\|_\infty \alpha + \tilde{\sigma}^2]
\end{aligned}$$

Therefore, there exists a universal constant $C > 0$ such that $K_0 \leq C\xi$ and with similar arguments we can show that $K_L \leq C\xi$ and $K_R \leq C\xi$. Which concludes that with probability at least $1 - 3\rho$,

$$\left| L_n \left(A_k, \widehat{d}_{s(n)}, \widehat{w}_{s(n)} \right) - L_n \left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)} \right) \right| \leq 3C\xi$$

□

Proof of Lemma 2 Assume that Hypothesis 1 and 2 are satisfied, fix $\mathbf{x} \in [0,1]^p$ and Θ. Then, let us show by contradiction that for all $\xi > 0$, there exists $N \in \mathbb{N}^\star$ such that, with probability at least $1 - \rho$ for all $n \geq N$

$$L_n \left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)} \right) \leq \xi.$$

So, assume that there exists $c > 0$, $0 < p_0 < 1$ and a sub-sequence $\phi(n)$ such that

$$L_{\phi(n)} \left(A_{s(\phi(n))}, \widehat{d}_{s(\phi(n))}, \widehat{w}_{s(\phi(n))} \right) > c$$

with probability at least p_0. To keep the notation simple, we omit to write $\phi(n)$ and still write n for the indexes of the sub-sequence. Additionally, assume that k is sufficiently large so that the conclusion of Technical Lemma 3 is satisfied, hence

$$\left|L_n\left(A_k, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) - L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right)\right| \leq \xi \tag{3.8}$$

and Equation (3.5) is satisfied. Note that all the feasible cuts d in A_k must be performed in $A_k \setminus A_{s(n)}$, otherwise d would split $A_{s(n)}$ and $(A_k)_k$ would not converge to $A_{s(n)}$ (see Figure 3.1 for an illustration in $p = 2$). From here we conclude that

$$L_n\left(A_k, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \leq \sup_{\substack{d \in \mathcal{C}_{A_k} \cap \mathcal{C}_{A_{s(n)}} \\ w \in \mathcal{W}_{A_k}}} L_n\left(A_k, d, w\right) \leq \sup_{\substack{d \in \mathcal{C}_{A_k} \\ w \in \mathcal{W}_{A_k}}} L_n\left(A_k, d, w\right) \leq \xi$$

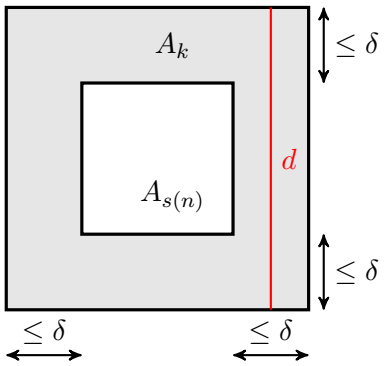

Figure 3.1: All feasible cuts in A_k must be performed in the $A_k \setminus A_{s(n)}$, like the cut d in the figure, otherwise the cut would split $A_{s(n)}$.

On the other hand, from Equation (3.8), with probability at least p_0, we have

$$c - \xi \leq L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) - \xi \leq L_n\left(A_k, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right)$$

Hence, we have, with probability at least p_0,

$$c - C\xi \leq L_n\left(A_k, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \leq \xi$$

which is absurd, since $c > 0$ is fixed and ξ is arbitrarily small. Thus the result follows.

□

3.4 Asymptotically the Regression Function has no Variation on Final Nodes

For a cell A, fix a cut $d \in \mathcal{C}_A$ and consider a function $w \in \mathcal{W}$, we need to define $L_n(A, d, w)$. This is done according to the following procedure, first create a random vector W of dimension $\widehat{N}^{(h)}_{miss}(A) = \text{Card}(\mathbf{i}^{(h)}_{A,miss})$, where $W_k = \text{Ber}(w(Y_{j_k}))$ for $j_k \in \mathbf{i}^{(h)}_{A,miss}$, then assign the observations $\widehat{\mathbf{X}}_{j_k}$ to the child nodes according to the random

vector W. Once we have assigned the observations to the child nodes, we evaluate the empirical CART criterion L_n considering these assignations. Note that in this case L_n is a random variable and the assignations are independent to each other so $L_n(A, d, w)$ is a sum of independent random variables with the same distribution. Hence, by the strong law of large numbers $L_n(A, d, w) \to L^\star(A, d, w)$ almost surely as $n \to \infty$ for all cuts $d \in \mathcal{C}_A$ and all functions $w \in \mathcal{W}$.

Proof of Proposition 1 With this extra definitions we can prove Proposition 1, that is, we can prove the almost sure convergence of $\Delta(m, A_{s(n)})$ towards 0 by showing that the theoretical CART criterion of the sequence $(A_{s(n)})_n$ tends to 0 and making use of Lemmas 1 and 2. Note that

$$\begin{aligned}
&L^\star\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) - L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \\
&= L^\star\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) - L_n\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) \\
&\qquad\qquad + L_n\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) - L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \\
&\leq L^\star\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) - L_n\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right)
\end{aligned}$$

Where the last inequality comes from noting that $L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \geq L_n\left(A_{s(n)}, d, w\right)$ for all cut $d \in \mathcal{C}_{A_{s(n)}}$ and assignation $w \in \mathcal{W}^{(\widehat{h})}_{A_{s(n)}}$, where $\widehat{d}_{s(n)} = (\widehat{h}, \widehat{z})$. As discussed above, by strong law of large numbers $L^\star\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) - L_n\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) \to 0$, almost surely. Fix $\xi, \rho > 0$, for n sufficiently large, we have

$$L^\star\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) - L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \leq \xi \quad \text{almost surely.}$$

On the other hand, by Lemma 2, there exists N_1 such that for all $n \geq N_1$, with probability at least $1 - \rho$

$$L_n\left(A_{s(n)}, \widehat{d}_{s(n)}, \widehat{w}_{s(n)}\right) \leq C\xi$$

Hence, with the same probability,

$$L^\star\left(A_{s(n)}, d^\star_{s(n)}, w^\star_{s(n)}\right) \leq \xi$$

And by Lemma 1, we conclude that

$$\Delta(m, A_{s(n)}) \xrightarrow{a.s.} 0$$

□

We are now ready to prove Theorem 1.

Proof of Theorem 1 Let

$$m_n(\mathbf{X}) = \frac{1}{\widehat{N}(A_{s(n)}(\mathbf{X}))} \sum_{i=1}^{n} Y_i \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}(\mathbf{X})}$$

our tree estimator and define two other quantities. The first one takes our partition of the cells $A_{s(n)}$ built up using the imputed variables but considers the local means using the complete (unseen) observations $\mathbf{X}_i$,

$$m'_n(\mathbf{X}) = \frac{1}{N(A_{s(n)}(\mathbf{X}))} \sum_{i=1}^{n} Y_i \mathbb{1}_{\mathbf{X}_i \in A_{s(n)}(\mathbf{X})},$$

while the second one takes $m(\mathbf{X}_i)$ for the prediction and the complete observations $\mathbf{X}_i$,

$$m''_n(\mathbf{X}) = \frac{1}{N(A_{s(n)}(\mathbf{X}))} \sum_{i=1}^{n} m(\mathbf{X}_i) \mathbb{1}_{\mathbf{X}_i \in A_{s(n)}(\mathbf{X})}.$$

By Equation (3.3), we know that for all $\alpha, \xi > 0$ there exists $N \in \mathbb{N}^\star$, such that for all $n \geq N$,

$$\begin{aligned}
\mathbb{P}\left[|m'_n(\mathbf{X}) - m''_n(\mathbf{X})| \geq \alpha\right] &= \mathbb{P}\left[\left|\frac{1}{N(A_{s(n)}(\mathbf{X}))} \sum_{i=1}^{n} \Big(Y_i - m(\mathbf{X}_i)\Big) \mathbb{1}_{\mathbf{X}_i \in A_{s(n)}(\mathbf{X})}\right| \geq \alpha\right] \\
&= \mathbb{P}\left[\left|\frac{1}{N(A_{s(n)}(\mathbf{X}))} \sum_{i=1}^{n} \varepsilon_i \mathbb{1}_{\mathbf{X}_i \in A_{s(n)}(\mathbf{X})}\right|\right] \\
&\leq \xi
\end{aligned}$$

On the other hand, note that $m''_n(\mathbf{X}) = \sum_{i=1}^{n} W_{n,i}(\mathbf{X}) m(\mathbf{X}_i)$, where

$$W_{n,i}(\mathbf{X}) = \frac{1}{N(A_{s(n)}(\mathbf{X}))} \mathbb{1}_{\mathbf{X}_i \in A_{s(n)}(\mathbf{X})}$$

Then,

$$\begin{aligned}
\mathbb{E}[m''_n(\mathbf{X}) - m(\mathbf{X})]^2 &= \mathbb{E}\left[\sum_{i=1}^{n} W_{n,i}(\mathbf{X}) m(\mathbf{X}_i) - m(\mathbf{X})\right]^2 \\
&= \mathbb{E}\left[\sum_{i=1}^{n} \sqrt{W_{n,i}(\mathbf{X})} \sqrt{W_{n,i}(\mathbf{X})} (m(\mathbf{X}_i) - m(\mathbf{X}))\right]^2 \\
&\quad \text{(Applying Cauchy-Schwartz's inequality)} \\
&\leq \mathbb{E}\left[\sum_{i=1}^{n} W_{n,i}(\mathbf{X}) \sum_{i=1}^{n} W_{n,i}(\mathbf{X}) (m(\mathbf{X}_i) - m(\mathbf{X}))^2\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n} W_{n,i}(\mathbf{X}) (m(\mathbf{X}_i) - m(\mathbf{X}))^2 \mathbb{1}_{\mathbf{X}_i \in A_{s(n)}(\mathbf{X})}\right]
\end{aligned}$$

Note that $(m(\mathbf{X}_i) - m(\mathbf{X}))^2 \mathbb{1}_{\mathbf{X}_i \in A_{s(n)}(\mathbf{X})} \leq \Delta(m, A_{s(n)}(\mathbf{X}))^2$, hence

$$\mathbb{E}[m''_n(\mathbf{X}) - m(\mathbf{X})]^2 \leq \mathbb{E}\left[\Delta(m, A_{s(n)}(\mathbf{X}))^2\right]$$

Since $\Delta(m, A_{s(n)}(\mathbf{X})) \leq \Delta(m, [0,1]^p) < \infty$, we can use the dominated convergence theorem, and conclude that, using Proposition 1,

$$\lim_{n \to \infty} \mathbb{E}[m''_n(\mathbf{X}) - m(\mathbf{X})]^2 = 0.$$

Hence we have the consistency $m'_n(\mathbf{X}) \xrightarrow{\mathbb{P}} m(\mathbf{X})$. This means that a (fictive) estimator built upon the empirical partition but where all the values are observed to compute the empirical mean step is consistent. We will use this fact to show that $m_n(\mathbf{X}) \xrightarrow{\mathbb{P}} m(\mathbf{X})$.

First, consider the case in dimension 1. We use the specific case where a cut d and an assignation w (of the cell $A_{s(n)}$) leaves all the observed points to the left and assigns all the missing observations to the right. By Lemma 2, we know that for n sufficiently large $L_n(A_{s(n)}, d, w) \leq \xi$. As already seen in Equation (1.1),

$$L_n(A_{s(n)}, d, w) = \frac{\widehat{N}_{obs}(A_{s(n)})\widehat{N}_{miss}(A_{s(n)})}{\widehat{N}(A_{s(n)})\widehat{N}(A_{s(n)})}\left(\widehat{Y}_{obs} - \widehat{Y}_{miss}\right)^2$$

By the convergence of $m'_n(\mathbf{X})$, we have, in particular that $\widehat{Y}_{obs} \xrightarrow{\mathbb{P}} m(\mathbf{X})$.

Using Hypothesis 2 and the same ideas as in Lemma 2, since $p_n^{(h)} \to c^{(h)} < 1$, there exists $N \in \mathbb{N}^\star$ such that, with probability at least $1-\rho$ for all $n \geq N$, $\widehat{N}_{obs}(A_{s(n)})/\widehat{N}(A_{s(n)}) \geq c$, where $c > 0$ is a constant. On the other hand, if $\widehat{N}_{miss}(A_{s(n)})/\widehat{N}(A_{s(n)}) \xrightarrow{\mathbb{P}} 0$ then, trivially $m_n(\mathbf{X}) \xrightarrow{\mathbb{P}} m'_n(\mathbf{X})$ so let us consider the case $\widehat{N}_{miss}(A_{s(n)})/\widehat{N}(A_{s(n)}) \geq c'$, hence with probability at least $1-\rho$,

$$\left(\widehat{Y}_{obs} - \widehat{Y}_{miss}\right)^2 \leq \frac{\xi}{cc'}.$$

This shows that the random variable $\widehat{Y}_{obs} - \widehat{Y}_{miss}$ converges to 0 in probability. Since $\widehat{Y}_{obs} \xrightarrow{\mathbb{P}} m(\mathbf{X})$, we obtain that $\widehat{Y}_{miss} \xrightarrow{\mathbb{P}} m(\mathbf{X})$. Finally, using the following formula for $m_n(\mathbf{X})$

$$m_n(\mathbf{X}) = \frac{\widehat{N}_{obs}(A_{s(n)}(\mathbf{X}))}{\widehat{N}(A_{s(n)}(\mathbf{X}))}\widehat{Y}_{obs} + \frac{\widehat{N}_{miss}(A_{s(n)}(\mathbf{X}))}{\widehat{N}(A_{s(n)}(\mathbf{X}))}\widehat{Y}_{miss}$$

we conclude that $m_n(\mathbf{X}) \xrightarrow{\mathbb{P}} m(\mathbf{X})$. For dimension bigger than 1, denote again $Y^{(j)} = m_j(X^{(j)}) + \epsilon^{(j)}$, where $\epsilon^{(j)} \sim \mathcal{N}(0, \sigma^2/p)$ so that we have $Y = \sum_{j=1}^p Y^{(j)}$ and define

$$m_n^{(j)}(\mathbf{X}) = \frac{1}{\widehat{N}(A_{s(n)}(\mathbf{X}))}\sum_{i=1}^n Y_i^{(j)} \mathbb{1}_{\widehat{\mathbf{X}}_i \in A_{s(n)}(\mathbf{X})}.$$

Consider the cut in the direction 1 which leaves all the observations where $\mathbf{M}^{(1)} = 0$ to the left and assigns the observations where $\mathbf{M}^{(1)} = 1$ to the right. We denote $\widehat{Y}_{obs(1)}$ and $\widehat{Y}_{miss(1)}$ the respective (on the left and on the right) empirical means. By the arguments in dimension 1, we have that $\widehat{Y}_{obs(1)} - \widehat{Y}_{miss(1)} \to 0$ in probability. By definition,

$$\widehat{Y}_{obs(1)} - \widehat{Y}_{miss(1)} = \widehat{Y}^{(1)}_{obs} - \widehat{Y}^{(1)}_{miss} + \left(\sum_{j=2}^p \widehat{Y}^{(j)}_{obs(1)} - \sum_{j=2}^p \widehat{Y}^{(j)}_{miss(1)}\right).$$

Since the random variable $Y^{(j)}$ $(j \neq 1)$ is independent of the random variable $\widehat{\mathbf{X}}^{(1)}$ conditionally to $\widehat{\mathbf{X}} \in A_{s(n)}$, the distributions of the two sums on the right hand side are equal. Since each random variable $\widehat{Y}^{(j)}$ converges (see Technical Lemma 1) we conclude that the difference of the two sums converges in probability to 0. Hence, $\widehat{Y}^{(1)}_{obs} - \widehat{Y}^{(1)}_{miss} \to 0$ in probability. This finally shows that $m_n^{(1)}(\mathbf{X}) \xrightarrow{\mathbb{P}} m_1(\mathbf{X})$. Similarly, we show that for all $j \geq 1$, $m_n^{(j)}(\mathbf{X}) \xrightarrow{\mathbb{P}} m_j(\mathbf{X})$ and then $m_n(\mathbf{X}) \xrightarrow{\mathbb{P}} m(\mathbf{X})$ by summation of the p previous convergences which concludes the proof of the Theorem 1.

□

4 Use of Autoencoders for the Reconstruction of Missing Data

According to Costa et al. (2018), neural network-based methods have been increasingly used for missing data imputation. However, deep learning architectures especially designed for this purpose has not yet been explored to its full potential. In this chapter we introduce different families of autoencoders and show how they might be used for imputation goals. We present scenarios where they have been applied successfully, review some drawbacks and give some extensions to overcome these issues. At the end of the chapter we briefly discuss their use in other fields like recommender systems.

Remember that an autoencoder is a type of neural network that learns a representation for a data set. Each autoencoder is composed by, at least, three layers: the input layer, the hidden layer and the output layer, which can be divided into two parts: an encoder (from the input layer to the hidden layer), and a decoder (from the hidden layer to the output layer). The encoder maps an input vector $\mathbf{x}$ into a hidden representation $\mathbf{z}$ belonging to a latent space, through a nonlinear transformation $e_\theta(\mathbf{x})$. The resulting representation $\mathbf{z}$ is then mapped back to a vector $\mathbf{y}$ which has the same dimension than $\mathbf{x}$, where $\mathbf{y}$ is equal to a nonlinear transformation $d_\phi(\mathbf{z})$. The training of an autoencoder consists in optimizing the model parameters θ and ϕ. Denoising Autoencoders (DAEs) are a variant of autoencoders that are designed to recover the original input from a noisy data $\tilde{\mathbf{x}}$ which can exist due to data corruption via some noise-additive mechanisms or by missing data. Another variant of the autoencoders are the Variational Autoencoders (VAEs) which take advantage of variational inference to create well organized latent space. This latent space allows the use of VAEs as generative models to not only recover the original input but to create new unseen observations similar to those belonging to the training data set. By imposing some changes to the loss function use to train VAEs, we show that they can be used for missing imputation.

4.1 Autoencoders

If we denote respectively as $\mathcal{E}$ and $\mathcal{D}$ the families of encoders and decoders we are considering, then the learning process is described simply as minimizing a loss function over these families, that is

$$(e^\star, d^\star) \in \underset{(e,d)\in\mathcal{E}\times\mathcal{D}}{\arg\min}\ L(\mathbf{x}, d(e(\mathbf{x})))$$

As in the case of supervised learning we have a training data set $\mathcal{D}_n = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$, so we aim to minimize the empirical risk, that is

$$(e^\star, d^\star) \in \underset{(e,d)\in\mathcal{E}\times\mathcal{D}}{\arg\min}\ \mathbb{E}_{\mathcal{D}_n} L(\mathbf{x}, d(e(\mathbf{x})))$$

Accordingly to Goodfellow et al. (2016), nearly all of deep learning is trained with stochastic gradient descent. Note that the empirical risk takes the form of an additive loss function, therefore the gradient descent algorithm requires to compute

$$\frac{1}{n}\sum_{i=1}^{n} \nabla L(\mathbf{x}_i, d(e(\mathbf{x}_i)))$$

As the training set size grows the time to take a single gradient step becomes larger, to overcome this problem we can sample a *minibatch* of examples $(\mathbf{x}_{j_1}, \ldots, \mathbf{x}_{j_m})$ drawn uniformly from the training set $\mathcal{D}_n$. The minibatch size m is chosen to be small and is usually held fixed even while the training set size n increases so we can reduce the time to calculate a step of the gradient descent. This algorithm is known as stochastic gradient descent.

Artificial neural networks consist of various layers of interconnected neurons powered by activation functions. Each neuron receives a multidimensional input, the neuron creates a linear combination of the input and adds a bias value, then this value is passed to an activation function which calculates the final value given out of the neuron. The parameters of the neural networks are the weight of the linear combinations and the bias value for each neuron while an activation function is basically just a simple function that transforms the inputs into outputs in a certain range of values. There are various types of activation functions that perform this task in a different manner. These activation functions introduce non-linear properties into the neural networks allow them to learn complex patterns in the data. We now define the activation functions that we are going to use throughout this work, consider $\mathbf{x} \in \mathbb{R}$, and let $\mathbf{w}$ and b the weight for the linear combination and the bias, these activation functions are:

- Linear activation: $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b$
- Exponential activation: $f(\mathbf{x}) = \exp\{\mathbf{w}^T\mathbf{x} + b\}$
- Sigmoid activation: $f(\mathbf{x}) = \frac{1}{1+e^{(-\mathbf{w}^T\mathbf{x}+b)}}$
- ReLU activation: $f(\mathbf{x}) = \max\{0, \mathbf{w}^T\mathbf{x} + b\}$

Linear output layers are often used to produce outputs that might take values in a range of $(-\infty, \infty)$, for example they can be used to model the mean of a Gaussian distribution. They represent little difficulty for gradient-based optimization algorithms. When we desire to have positive values for the output, for example when we like to model the variance of a distribution, we can use the exponential function. The sigmoid function is usually used when we are interested to model a probability since the range of values are between $(0, 1)$. In modern neural networks it is common to use the rectified linear unit (ReLU) activation function as a way to introduce non-linearity into the network (Jarrett et al., 2009; Nair and Hinton, 2010; Glorot, Bordes, and Bengio, 2011), becoming the most popular activation function for neural networks (Ramachandran, Zoph, and Le, 2017). It is not the intention of this work to introduce several activation functions explored in the literature, but only to describe those that we are going to use. For a more extensive review on activation functions, their properties and limitations we refer to Goodfellow et al. (2016).

4.1.1 Example (MNIST Data Set)

If x_j follows a Bernoulli distribution we could choose $L(x_j, y_j)$ as the negative of the log-likelihood, being y_j the parameter of such distribution and take the average over these loss functions, that is

$$L(\mathbf{x}, \mathbf{y}) = \frac{1}{p}\sum_{j=1}^{p} L(x_j, y_j) = \frac{1}{p}\sum_{j=1}^{p} -x_j \log(y_j) - (1 - x_j)\log(1 - y_j)$$

where $\mathbf{y} = d(e(\mathbf{x}))$, and e and d are neural networks trained with stochastic gradient descent. Note that x_j can only take the values 0 or 1, while y_j can take any value in the continuous $[0, 1]$, hence as a final reconstruction of x_j we can take $\hat{x}_j = \mathbb{1}_{y_j \geq 0.5}$. Taking $L(x_j, y_j)$ in this way corresponds with the binary cross-entropy whose minimization coincides with minimizing the Kullback-Leibler divergence and thus attempts to find the value of y_j that best approximates x_j (Goodfellow et al., 2016).

We are going to use the MNIST data to illustrate the different ideas throughout this chapter. The MNIST data sets consist of images of 28×28 gray scale pixels (hence, each image has 784 pixels). Where each image is a hand-written integer between 0 and 9. The training data set has 60,000 images and the testing data set has 10,000 images.

We have binarized the images, assigning 1 if the value of the pixel is bigger or equal to 0.5 and assigning 0 otherwise, so we can maximize the Bernoulli log-likelihood for each pixel. The encoder-decoder structure is symmetric, where e and d are multilayer perceptrons, e has a hidden layer of 392 neurons followed by another hidden layer of 196 neurons, we use the ReLU activation function for all the hidden layers, except for the code $\mathbf{z}$ where we use a linear activation function. The last layer of the decoder has 784 neurons and a sigmoid activation function. Finally, we reconstruct the binarized images applying the function $\hat{x}_j = \mathbb{1}_{y_j \geq 0.5}$. We vary the dimension of the code k to be 2 or 98. Figure 4.1[1] shows a diagram of the autoencoder.

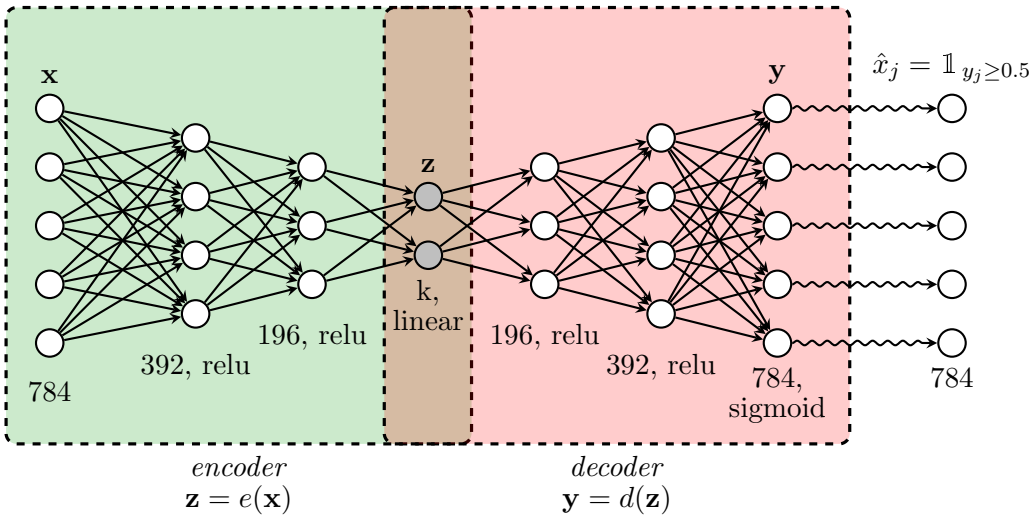

Figure 4.1: Deep autoencoder used in our experiments, we vary the dimension of the latent space k to be 2 or 98.

[1]The representation diagrams for autoencoders are based on the work of Petar Veličković which can be found here: https://github.com/PetarV-/TikZ/tree/master/Variational denoising autoencoder

Figures 4.2 and 4.4 show the reconstruction of the first 10 images of the testing data set (once the autoencoder has learned with the training data set) when the dimension of the latent space is $k = 2$ and $k = 98$, respectively. For comparison, we have performed PCA, Figure 4.3 and Figure 4.5 show the reconstruction of the same images using the first 2 principal components and the first 98 principal components, respectively. In Figure 4.6 and Figure 4.7 we compare the latent space induced by our autoencoder when $k = 2$ and the space induced by the first 2 principal components.

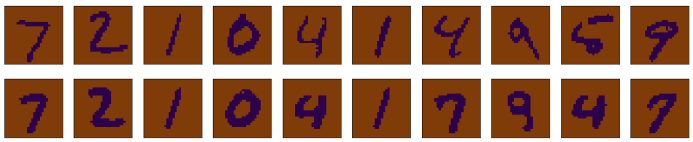

Figure 4.2: Reconstruction of the first 10 images of the testing MNIST data set, using our autoencoder when the dimension of the latent space is $k = 2$.

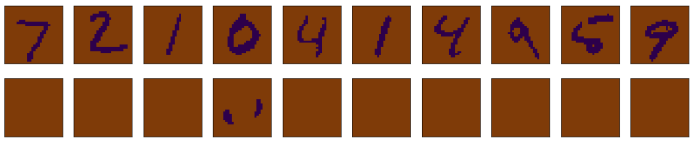

Figure 4.3: Reconstruction of the first 10 images of the testing MNIST data set, using the first 2 principal components.

We observe in Figures 4.2 and 4.3, how the autoencoder is able to reconstruct the the inputs into images that fairly look as numbers (even while some of them do not resemble the same number that is shown in the input), even when all the information has been collapsed into a 2 dimensional latent space. On the other hand the linear reduction of dimensionality performed by PCA is unable to preserve the structure of the inputs so it could not reconstruct them into images that look like numbers, in fact most of the outputs are just "blank" images.

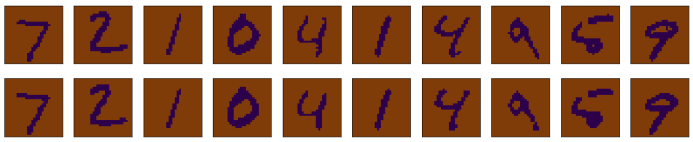

Figure 4.4: Reconstruction of the first 10 images of the testing MNIST data set, using our autoencoder when the dimension of the latent space is $k = 98$.

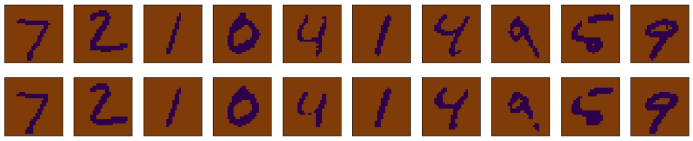

Figure 4.5: Reconstruction of the first 10 images of the testing MNIST data set, using the first 98 principal components.

We observe in Figures 4.4 and 4.5 that when we allow the latent space to have 98 dimensions, both the autoencoder and PCA are able to preserve important characteristics of the inputs so they can reconstruct (almost perfectly) the original images.

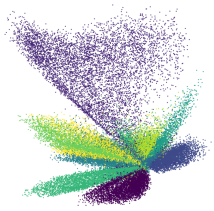

Figure 4.6: Latent space of the MNIST data set when the dimension of the latent space is $k = 2$, the color of each point correspond to the number represented in the input image.

Figure 4.7: Induced space by the first two principal components of the MNIST data set, the color of each point correspond to the number represented in the input image.

We observe in Figures 4.6 and 4.7 how the nonlinearity and the depth introduced in the architecture of the autoencoder give them an important improvement over the dimensionality reduction performed by PCA. These improvements allow the autoencoder to distinguish the numbers represented by the data set in the latent space while PCA is unable to distinguish between the different integers represented in the MNIST data sets.

4.2 Missing Data with Denoising Autoencoders

A denoising autoencoder is an autoencoder that receives a corrupted data point as its input and is trained to predict the original, uncorrupted data point as its output. These autoencoders assume that the original training data is clean, although corrupted instances are expected during the testing phase.

Traditionally, autoencoders minimize some function

$$L(\mathbf{x}, d(e(\mathbf{x})))$$

where L is a loss function penalizing $d(e(\mathbf{x}))$ for being dissimilar from $\mathbf{x}$. This encourages $d \circ e$ to be merely the identity function if they have the capacity to do so. As the name suggests, denoising autoencoder is an autoencoder having the denoising property. Thus, a denoising autoencoder minimizes

$$L(\mathbf{x}, d(e(\tilde{\mathbf{x}})))$$

where $\tilde{\mathbf{x}}$ is a copy of $\mathbf{x}$ that has been corrupted. Denoising autoencoders must therefore undo this corruption rather than simply copy their input. They were first introduced by Vincent et al. (2008) as a robust procedure to get a good representation of the input data. The idea is that such a good representation should capture the structure of the useful features and regularity characteristics of the distribution of its observed input making possible to reconstruct it from partial observation only. From an input observation $\mathbf{x}$ a corrupted version $\tilde{\mathbf{x}}$ is obtained applying a corruption process $p(\tilde{\mathbf{x}}|\mathbf{x})$. This corruption process corresponds to noise injection which should be calibrated to the nature of the input and the phenomenon that makes the testing data set to be corrupted (see Figure 4.8 for a diagram of a denoising autoencoder with a single hidden layer). Corrupting the original input $\mathbf{x}$ into $\tilde{\mathbf{x}}$ by adding small noise or forcing a fraction of elements in $\mathbf{x}$ to some default values makes the model to become more robust and to avoid overfitting. Some kind of noise that can be added are:

- Gaussian noise: This type of noise can be used for real valued inputs. The noise is added accordingly with a Gaussian distribution with mean zero and variance λ which acts as a regularization parameter.
- Masking noise: When we consider binary inputs $\{0, 1\}$, we can set a fraction of points λ to zero.
- Salt-and-pepper noise: Similarly to the masking noise a fraction of points λ are set to either the maximum or the minimum possible values according to a fair coin flip.

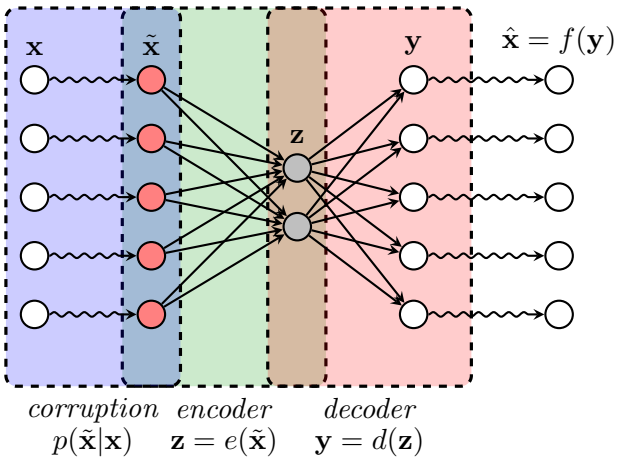

Figure 4.8: A shallow denoising autoencoder. The corruption process is usually a procedure that introduces some noise to the original data. Reconstructing the original input from a corrupted copy forces the autoencoder to get a robust representation of the input data in its latent space $\mathbf{z}$.

As pointed out by several authors (Ryu, Kim, and Kim, 2020; Ma et al., 2020; Gondara and Wang, 2018; Costa et al., 2018; Vincent et al., 2008), denoising autoencoders can be used to impute missing values. Instead of adding noise to the input and use denoising autoencoders, the corruption process replaces the missing values on the data with a predetermined value. If we denote by $\mathbf{m}$ the missing pattern, then the corruption process $p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})$ represents the conditional distribution over corrupted

samples given the original data sample and the missing pattern. Figure 4.9 shows a diagram for the use of a denoising autoencoder with missing data. During the training of a denoising autoencoder to impute missing values it is important to count with complete observations and to know the missing pattern and the missing mechanism which can be inferred from the matrix of missingness (Ma et al., 2020), an improper choice of the mechanism of missingness can bias the model (Costa et al., 2018). Then, we can replicate this mechanism in the complete observations so we can try to reconstruct the original input with the denoising autoencoder.

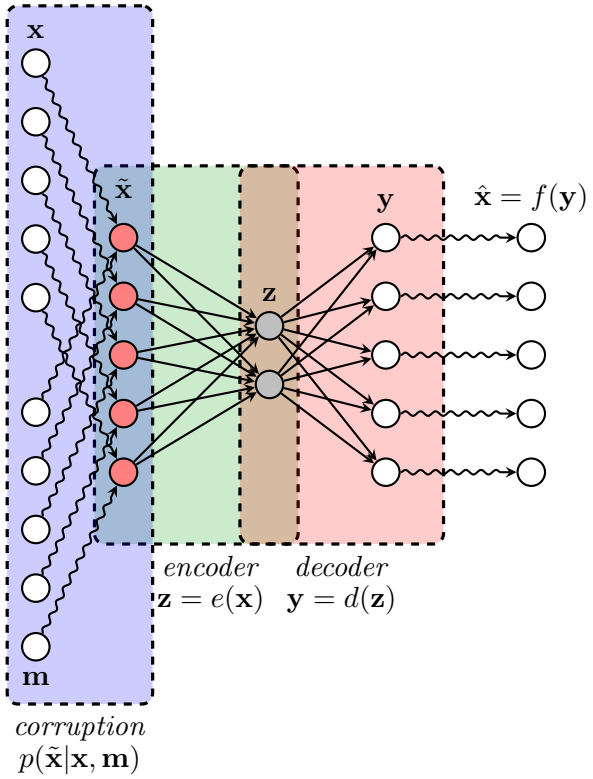

Figure 4.9: A shallow denoising autoencoder used to impute missing data. The corruption process corresponds to the replacement of the missing values with some predetermined value.

4.2.1 Example (MNIST Data Set, Continuation)

Continuing with our example, we introduce missing values to the MNIST data set. We vary the number of pixels with missing values to be 157 pixels or 706, which are approximately 20% and 90% of the image, respectively, where the pixels to be missing are selected completely at random. Remember that previously we have binarized the images, so each pixel takes the values 0 or 1, those pixels whose value is missing are set as 0.5, so the corruption process is given as

$$\tilde{x}_j|\mathbf{x},\mathbf{m} = \begin{cases} 1 & \text{if } x_j = 1 \text{ and } m_j = 0 \\ 0 & \text{if } x_j = 0 \text{ and } m_j = 0 \\ 0.5 & \text{if } m_j = 1 \end{cases} \tag{4.1}$$

The architecture of the autoencoder is the same as before, that is, we use the same number of layers and neurons and the same activation functions (Figure 4.1 shows a

diagram of the autoencoder) and vary the dimension k of the latent space to be 2 or 98, as before. Figures 4.10 and 4.11 show the reconstruction of the first 10 images of the testing MNIST data set when there are 157 pixels with missing values, $k = 2$ and $k = 98$, respectively. The reconstruction of the same images when 706 pixels are missing are shown in Figures 4.12 and 4.13, $k = 2$ and $k = 98$, respectively. The original image is shown at the top, the image with missing values is shown in the middle and the reconstruction done by the denoising autoencoder is shown at the bottom. The latent space when $k = 2$ is similar to the one shown in Figure 4.6 and is omitted.

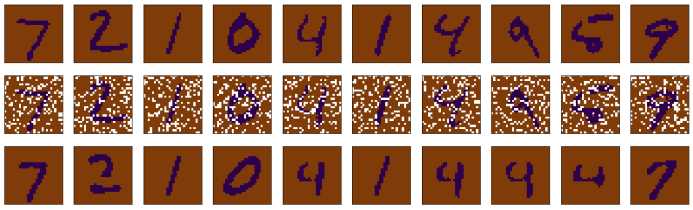

Figure 4.10: Reconstruction of the first 10 images of the test MNIST data set, using our denoising autoencoder when the dimension of the latent space is $k = 2$ and 157 pixels have missing values.

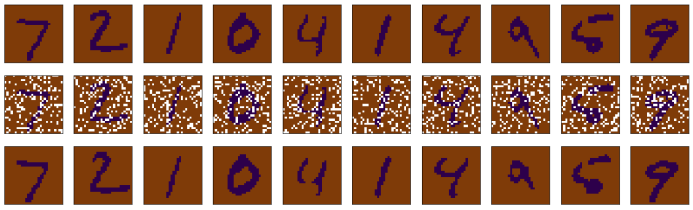

Figure 4.11: Reconstruction of the first 10 images of the test MNIST data set, using our denoising autoencoder when the dimension of the latent space is $k = 98$ and 157 pixels have missing values.

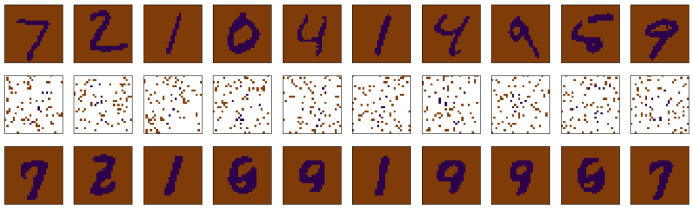

Figure 4.12: Reconstruction of the first 10 images of the test MNIST data set, using our denoising autoencoder when the dimension of the latent space is $k = 2$ and 704 pixels have missing values.

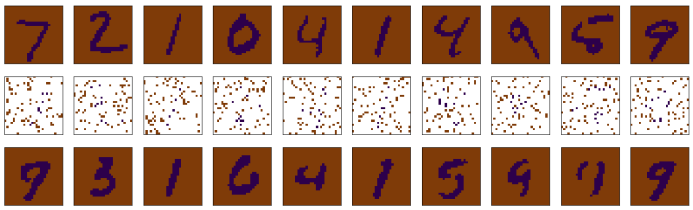

Figure 4.13: Reconstruction of the first 10 images of the test MNIST data set, using our denoising autoencoder when the dimension of the latent space is $k = 98$ and 704 pixels have missing values.

4.2.2 Extensions of Denoising Autoencoders

Denoising autoencoders were designed to learn a good representation of the data, rather than performing imputation of missing values which, according to Ma et al. (2020), leads to suboptimal results when we apply directly denoising autoencoders for imputation of missing values since the loss function considers to recover the entire original input instead of recover the missing values only. Therefore the same authors consider a loss function that only attempts to recover the missing values

$$\mathbb{E}_{\mathcal{D}_n}\left[L(\mathbf{m} \circ \mathbf{x}, d(e(\mathbf{m} \circ \tilde{\mathbf{x}})))\right]$$

where $\circ$ indicates element-wise product. Ryu, Kim, and Kim (2020) compares DAEs, VAEs (this kind of autoencoders are defined and studied in Sections 4.4 and 4.5) and Wasserstein Autoencoders (Tolstikhin et al., 2017) in an imputation study with time series data, concluding that the proposed denoising autoencoder shows lower imputation errors compared to the other two autoencoders. Furthermore, in the same spirit of getting more attention to the imputation of the missing values rather than the reconstruction of the whole input, their loss function is of the form

$$\mathbb{E}_{\mathcal{D}_n}\left[\alpha L(\mathbf{m} \circ \mathbf{x}, d(e(\mathbf{m} \circ \tilde{\mathbf{x}}))) + (1-\alpha) L(\tilde{\mathbf{m}} \circ \mathbf{x}, d(e(\tilde{\mathbf{m}} \circ \tilde{\mathbf{x}})))\right]$$

where $\tilde{\mathbf{m}}$ is the complement of $\mathbf{m}$, that is $\tilde{m}_j = 1$ if x_j is observed and zero otherwise, large values of α would encourage the autoencoder to give more importance to the imputation of the missing values rather than the reconstruction of the full input (the authors consider $\alpha = 0.8$).

A Stacked Denoising Autoencoder (SDAE) (Vincent et al., 2010) is a special case of denoising autoencoder in which the activation of an input is corrupted before passing to next hidden layer which in turn tries to reconstruct the original activation value of the previous layer. Costa et al. (2018) perform a comparison study between SDAEs and other techniques to impute like Support Vector Machine (SVM) imputation (Mallinson and Gammerman, 2003), Multiple Imputation by Chained Equations (MICE) (Van Buuren et al., 2006; Azur et al., 2011) , k Nearest Neighbors (kNN) imputation (for more details on kNN see Troyanskaya et al. (2001), Xia et al. (2017), and García-Laencina, Sancho-Gómez, and Figueiras-Vidal (2013)), Mean imputation or EM (Dempster, Laird, and Rubin, 1977) imputation, concluding that SVM imputation ensures the best performance while MICE perform better in terms of imputation quality. Gondara and Wang (2018) also studied SDAEs for multiple imputation compared with MICE on MCAR and MNAR mechanisms, using sum of Root Mean Squared Error as the performance

metric. In a second work Gondara and Wang (2017) propose a SDAE model to handle imputation in healthcare data. The simulation results showed advantages on the use of SDAE for imputation. Beaulieu-Jones and Moore (2017) use SDAE to impute data in electronic health records, comparing this approach to other five imputation strategies (including mean imputation, *knn*-imputation and MICE). The results show that the proposed SDA approach outperforms the other methods. Ning et al. (2017) proposed an algorithm based on SDAE which is compared to other two imputation algorithms based on the *knn* algorithm, showing that the proposed imputation method outperforms the ones used for comparison. The above works show that deep learning techniques are promising in the field of imputation.

The uses of denoising autoencoders are not limited on denoising images or missing imputation, but also as a procedure to initialize a neural network (Vincent et al., 2008; Xie, Xu, and Chen, 2012; Erhan et al., 2010), which can be then trained to accomplish a supervised learning task. The idea is that a neural network that can extract a good representation of the input data must be also able to accomplish a supervised task. In this case the architecture of the neural network use for the supervised task is copied in the encoder, whose trained parameters are used as initial values to train the neural network in the supervised learning task. In this spirit, Xie, Xu, and Chen (2012) uses Stacked Sparse Denoising Autoencoders (SSDAE) as a way to initialize a second denoising autoencoder use to denoise images (their final goal). To induce this sparsity in the denoising autoencoders they introduce a sparsity-inducing term, so their loss function takes the form

$$\mathbb{E}_{\mathcal{D}_n} L(\mathbf{x}, d(e(\tilde{\mathbf{x}}))) + \beta D_{KL}(\hat{\rho}||\rho) + \frac{\lambda}{2}(\|\mathbf{W}\|_F^2 + \|\mathbf{W}'\|_F^2)$$

where

$$D_{KL}(\hat{\rho}||\rho) = \sum_j^{|\hat{\rho}|} \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}, \quad \hat{\rho} = \mathbb{E}_{\mathcal{D}_n} p(\tilde{\mathbf{x}}|\mathbf{x})$$

$\mathbf{W}$ and $\mathbf{W}'$ are the weights of the encoder and the decoder (resp.), and β, λ, ρ are hyper-parameters defined by the user. According to the authors, sparse variants of deep neural networks are expected to perform well in vision problems because they have a similar structure to human visual cortex (see also Lee, Ekanadham, and Ng (2007)).

According to Bengio et al. (2013) the first ideas about the probabilistic interpretation of autoencoders were proposed by Ranzato, Boureau, and LeCun (2008) while Vincent (2011) gave the first formal probabilistic interpretation of regularized autoencoders. Alain and Bengio (2014) generalized the result of Vincent (2011) showing that when a denoising autoencoder is trained with small Gaussian noise and squared error loss, it estimates the score (derivative of the log-likelihood) of the underlying data-generating distribution. This led to proposals for sampling from the implicitly learned density function (see for example Rifai et al. (2012)). Bengio et al. (2013) showed that if an observation $\mathbf{x}$ is corrupted using the conditional distribution $p(\tilde{\mathbf{x}}|\mathbf{x})$ then the denoising autoencoder estimates the reverse conditional $p(\mathbf{x}|\tilde{\mathbf{x}})$, and showed that it is possible to recover a consistent estimator of $p(\mathbf{x})$ by iterative sampling through a Markov chain between $p(\mathbf{x}|\tilde{\mathbf{x}})$ and $p(\tilde{\mathbf{x}}|\mathbf{x})$. A breakthrough on generative models came with the arrival of variational autoencoders (Kingma and Welling, 2013) which explicitly model the encoder and decoder as conditional densities and train them making use of the variational inference procedure, the subject of the next section.

4.3 Variational Inference

Variational inference is a method for approximating probability densities (Jordan et al., 1999). The main idea is to use optimization, assume that we have a joint density of latent variables $\mathbf{z}$ and observed variables $\mathbf{x}$, $p(\mathbf{x}, \mathbf{z})$. The inference problem is to compute the conditional density of the latent variables given the observations $p(\mathbf{z}|\mathbf{x})$, this conditional density can be written as

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{z})}{p(\mathbf{x})}$$

The denominator is the density of the observations, also called the evidence, which can be calculated marginalizing out the latent variables from the joint density,

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p(\mathbf{z}) p(\mathbf{x}|\mathbf{z}) d\mathbf{z}$$

where $p(\mathbf{z})$ is the prior density for the latent variables and $p(\mathbf{x}|\mathbf{z})$ is the likelihood of the observations. However, it is quite common that this integral is intractable for moderately complicated likelihoods. Making intractable to produce inferences from the posterior density $p(\mathbf{z}|\mathbf{x})$. The idea of variational inference is to propose a family $\mathcal{Q}$ of densities over the latent variable and then find the member of the family that minimizes the Kullback-Leibler (KL) divergence to the exact posterior,

$$q^\star \in \arg\min_{q \in \mathcal{Q}} D_{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x})). \tag{4.2}$$

Thus, variational inference turns the inference problem of the posterior distribution into an optimization problem. The complexity of the family considered $\mathcal{Q}$ determines the complexity of the optimization, being more difficult to optimize over a complex family than a simple one. Hence, $\mathcal{Q}$ is chosen to be flexible enough to have a member close to $p(\mathbf{z}|\mathbf{x})$, but at the same time simple enough to allow an efficient optimization.

Equation (4.2) cannot be computed since it requires to calculate the evidence $\log p(\mathbf{x})$, which we have assume to be intractable. Fortunately, we can derive an equivalent expression which enables us to solve the optimization problem. To see this, expand the KL divergence presented in Equation (4.2),

$$\begin{aligned} D_{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{\mathbf{z}\sim q}\left[\log q(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q}\left[\log q(\mathbf{z}|\mathbf{x})\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}, \mathbf{z})\right] + \log p(\mathbf{x}) \end{aligned}$$

Let us define the evidence lower bound as

Definition 8 Evidence Lower Bound (ELBO)

$$\text{ELBO}(q) = \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}, \mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\log q(\mathbf{z}|\mathbf{x})\right] \tag{4.3}$$

then an equivalent optimization problem to Equation (4.2) is

$$q^\star \in \arg\max_{q \in \mathcal{Q}} \text{ELBO}(q)$$

Furthermore, note that $\log p(\mathbf{x})$ might be written as

$$\log p(\mathbf{x}) = D_{KL}(q(\mathbf{z}|\mathbf{x}) || p(\mathbf{z}|\mathbf{x})) + \text{ELBO}(q) \tag{4.4}$$

and since $D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) \geq 0$ then

$$\log p(\mathbf{x}) \geq ELBO(q)$$

so as it names suggests the ELBO is a lower bound for the evidence.

Applying simple algebra to Equation (4.3) we can rearrange ELBO(q) to get a more convenient form

$$\begin{aligned} \text{ELBO}(q) &= \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x},\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\log q(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}|\mathbf{z})\right] + \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q}\left[\log q(\mathbf{z}|\mathbf{x})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{\text{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \end{aligned} \tag{4.5}$$

Note from Equation (4.5) that mazimizing ELBO(q) implies a trade-off between maximizing the expected log-likelihood and at the same time minimizing the KL divergence between the approximate posterior $q(\mathbf{z})$ and the prior $p(\mathbf{z})$. This is a well-known trade-off that appears naturally in Bayesian inference problems and expresses the balance that needs to be found between the confidence we have in the data and the confidence we have in the prior.

Consider again Equation (4.4), so we can write the ELBO as

$$\text{ELBO}(q) = \log p(\mathbf{x}) - D_{KL}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x}))$$

while this expression cannot be computed, it is useful from a theoretical perspective. For example, it is known that the variational inference tends to underestimate the variance of the posterior density, this is (at least partially) for considering the KL divergence, since this divergence penalizes placing mass in q on areas where the posterior has little mass, but penalizes less the inverse. However, some empirical research have shown that variational inference does not necessarily suffer in accuracy (Blei, Jordan, et al., 2006; Braun and McAuliffe, 2010; Kucukelbir et al., 2017).

When we consider the KL divergence as the base of the optimization the variational inference is also called Kullback-Leibler variational inference (Barber, 2012). However, we can use different procedures to approximate the density through optimization, as it is emphasized by Wainwright and Jordan (2008). Some of these procedures could be the expectation propagation (Minka, 2001), belief propagation (Yedidia, Freeman, and Weiss, 2001), Laplace approximation (see, for example, Barber (2012)) or even developed divergences based on lower bounds that are tighter than the ELBO (Leisink and Kappen, 2001), as the variational cumulant expansion (Barber and Laar, 1999) or generalizations of the KL divergence as the $\alpha-$divergence which, in theory, provides better results than the KL divergence (Minka et al., 2005).

Note that the first term on the right hand side of Equation (4.3) corresponds to the expected complete log likelihood which is optimized by the EM algorithm (Dempster, Laird, and Rubin, 1977) when $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x})$, that is $Q(\theta|\theta^{(t)}) = \mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{x};\theta^{(t)})}\left[\log p(\mathbf{x},\mathbf{z};\theta)\right]$. This observation has led to another variety of variational inference algorithms (Waterhouse, MacKay, and Robinson, 1996; MacKay, 1997). Finally for further reading on variational inference, its comparison with MCMC, extensions and research on the field, applications, and open problems we recommend the review presented by Blei, Kucukelbir, and McAuliffe (2017) and the references therein.

4.4 Variational Autoencoders

Depending on the application on hand, after training an autoencoder it could be not necessary to use both the encoder and the decoder portions. For example, when the

goal is dimensionality reduction, we can use the encoder in order to create reduced representations of the data. The reconstruction of the decoder might not be required.

Traditional autoencoders are designed to encode and decode with as few loss as possible, no matter how the latent space is organized. Thus, traditional autoencoders could take advantage of any overfitting possibility to achieve this task. In order to be able to use the decoder of our autoencoder for generative purposes, we have to be sure that the latent space is regular enough. One way to do this is adding a regularization term to the loss function.

A variational autoencoder (VAE) emphasizes the role of the decoder portion of the network as a creative generator of new data points. In order to have an efficient generative process it is important than the code space is well organized. The regularity that is expected from the latent space in order to make the generative process possible can be expressed through two main properties: continuity (two close points in the latent space should not give two completely different contents once decoded) and completeness (a point sampled from the latent space should give meaningful content once decoded).

Equation (4.5) is the core and loss function of a VAE since $q(\mathbf{z}|\mathbf{x})$ is encoding $\mathbf{x}$ into $\mathbf{z}$ and $p(\mathbf{x}|\mathbf{z})$ is decoding it to reconstruct $\mathbf{x}$. We recognize $\mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}|\mathbf{z})\right]$ as the reconstruction log-likelihood found in other autoencoders, while $D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z}))$ forces the distribution returned by the encoder to be close to the prior $p(\mathbf{z})$ and thus regularizes the latent space. Therefore, the first term on the right hand side of Equation (4.5) is known as the "reconstruction term" and the second term is known as the "regularization term". Figures 4.19 and 4.20 show the latent space created by a traditional autoencoder and a VAE, respectively. We can observe how the VAE has introduced regularity to the latent space.

4.4.1 Bernoulli Likelihood

Assume that $\mathbf{z} \sim \mathcal{N}_k(0, I)$, that $\mathbf{x}|\mathbf{z} = (x_1|\mathbf{z}, \ldots, x_p|\mathbf{z})$ where $x_j|\mathbf{z} \overset{iid}{\sim} \mathrm{Ber}(d_j(\mathbf{z}))$ and $d = (d_1, \ldots, d_p)$ belongs to a family of functions $\mathcal{D}$. Let us consider, for now, that d is well defined and fixed. We are going to approximate $p(\mathbf{z}|\mathbf{x})$ by a Gaussian distribution $q(\mathbf{z}|\mathbf{x})$, whose mean and covariance matrix are defined by two functions of $\mathbf{x}$, $\mu(\mathbf{x})$ and $\sigma(\mathbf{x})$, that is $q(\mathbf{z}|\mathbf{x}) \equiv \mathcal{N}_k(\mu(\mathbf{x}), \sigma(\mathbf{x}))$. Furthermore, let us assume that the covariance matrix of $q(\mathbf{z}|\mathbf{x})$ is diagonal, so $\sigma(\mathbf{x}) = \mathrm{diag}(\sigma_1(\mathbf{x}), \ldots, \sigma_k(\mathbf{x}))$ and $\mu(\mathbf{x}) = (\mu_1(\mathbf{x}), \ldots, \mu_k(\mathbf{x}))$. These two functions belong, respectively, to families of functions $\mathcal{M}$ and $\mathcal{S}$.

Note that

$$\log p(\mathbf{x}|\mathbf{z}) = \sum_{j=1}^{p} x_j \log(d_j(\mathbf{z})) + (1 - x_j)\log(1 - d_j(\mathbf{z}))$$

Therefore, from Equation (4.5), we see that maximizing the ELBO over $q(\mathbf{z}|\mathbf{x})$ is equivalent to

$$q^\star \in \arg\max_{q\in\mathcal{Q}} \mathbb{E}_{\mathbf{z}\sim q}\left[\sum_{j=1}^{p} x_j \log(d_j(\mathbf{z})) + (1 - x_j)\log(1 - d_j(\mathbf{z}))\right] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{4.6}$$

Now, the second term on the criterion to maximize in (4.6) is the KL divergence between

two Gaussian distributions, which can be computed in closed form (Doersch, 2016) as

$$D_{\mathrm{KL}}(\mathcal{N}_k(\mu_0,\Sigma_0)||\mathcal{N}_k(\mu_1,\Sigma_1)) = \frac{1}{2}\Bigg\{ \mathrm{tr}\left(\Sigma_1^{-1}\Sigma_0\right) + (\mu_1-\mu_0)^T\Sigma_1^T(\mu_1-\mu_0) - k + \log\left(\frac{\det(\Sigma_1)}{\det(\Sigma_0)}\right)\Bigg\}$$

In our case, this simplifies to

$$\begin{aligned} D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) &= \frac{1}{2}\left\{\mathrm{tr}(\sigma(\mathbf{x})) + \|\mu(\mathbf{x})\|^2 - k - \log\det(\sigma(\mathbf{x}))\right\} \\ &= \frac{1}{2}\sum_{\kappa=1}^{k}\left[\sigma_\kappa(\mathbf{x}) + \mu_\kappa^2(\mathbf{x}) - \log\sigma_\kappa(\mathbf{x})\right] - \frac{k}{2} \end{aligned} \tag{4.7}$$

Substituting Equation (4.7) into (4.6), we get that maximizing the ELBO over q is equivalent to

$$\begin{aligned} (\mu^\star,\sigma^\star) \in \underset{(\mu,\sigma)\in\mathcal{M}\times\mathcal{S}}{\arg\max}\ & \mathbb{E}_{\mathbf{z}\sim q}\left[\sum_{j=1}^{p} x_j\log(d_j(\mathbf{z})) + (1-x_j)\log(1-d_j(\mathbf{z}))\right] \\ & - \frac{1}{2}\sum_{\kappa=1}^{k}\left[\sigma_\kappa(\mathbf{x}) + \mu_\kappa^2(\mathbf{x}) - \log\sigma_\kappa(\mathbf{x})\right] \end{aligned} \tag{4.8}$$

Up to now, we have assumed the function d known and fixed. In practice, however, the likelihood is not completely known and it is suitable to be maximized through its unknown parameters. This means that d is not fixed and needs to be learned. Thus, the ELBO is a function of both $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$ as

$$ELBO(q(\mathbf{z}|\mathbf{x}),p(\mathbf{x}|\mathbf{z})) = \mathbb{E}_{\mathbf{z}\sim q}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{4.9}$$

It is straightforward from Equation (4.8) that maximizing the ELBO over $q(\mathbf{z}|\mathbf{x})$ and $p(\mathbf{x}|\mathbf{z})$ is equivalent to

$$\begin{aligned} (d^\star,\mu^\star,\sigma^\star) \in \underset{(d,\mu,\sigma)\in\mathcal{D}\times\mathcal{M}\times\mathcal{S}}{\arg\max}\ & \mathbb{E}_{\mathbf{z}\sim q}\left[\sum_{j=1}^{p} x_j\log(d_j(\mathbf{z})) + (1-x_j)\log(1-d_j(\mathbf{z}))\right] \\ & - \frac{1}{2}\sum_{\kappa=1}^{k}\left[\sigma_\kappa(\mathbf{x}) + \mu_\kappa^2(\mathbf{x}) - \log\sigma_\kappa(\mathbf{x})\right] \end{aligned} \tag{4.10}$$

We have set a probabilistic model that depends on three functions, d, μ and σ, and express, using variational inference, the optimization problem to solve in order to get $d^\star$, $\mu^\star$ and $\sigma^\star$ that give the optimal encoding-decoding scheme with this model. As we cannot easily optimize over the entire space of functions, we constrain the optimization domain and decide to express d, μ and σ as neural networks. Hence, $\mathcal{D}$, $\mathcal{M}$ and $\mathcal{S}$ correspond respectively to the families of functions defined by the networks architectures and the optimization is done over the parameters of these networks. In practice, μ and σ are not defined by two completely independent networks instead they share part of their architecture and their weights (Rocca, 2019).

The neural networks, d, μ and σ are trained using stochastic gradient descent. Due to its simple, closed form, it is clear how to compute the gradient of the second

term on the right hand side of Equation (4.10). The first term on the right hand side of Equation (4.10) is a bit more tricky. The expectation $\mathbb{E}_{\mathbf{z}\sim q}[\log p(\mathbf{x}|\mathbf{z})]$ could be estimated using sampling, but getting a good estimate would require many samples of $\mathbf{z} \sim q$. However, we can take one sample as long as the minibatch size is large enough (Kingma and Welling, 2013). Figure 4.14 shows a general diagram of a VAE.

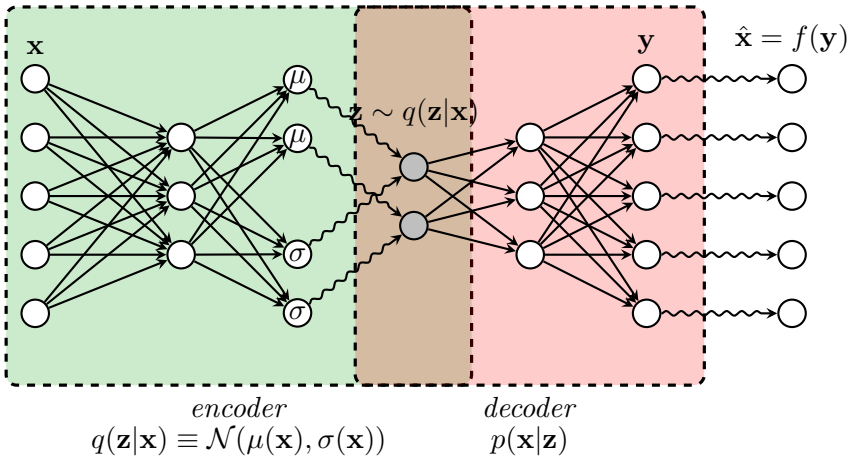

Figure 4.14: Diagram of a shallow VAE, note that we sample at the code layer $\mathbf{z}$. This sampling does not allow us to back propagate the error and avoid using stochastic gradient descent to train the autoencoder.

There is a significant problem in the variational autoencoder represented schematically in Figure 4.14. The forward pass of this autoencoder works fine and, if the output is averaged over many samples of $\mathbf{x}$ and $\mathbf{z}$, produces the expected value. However, we need to back-propagate the error through a layer that samples $\mathbf{z}$ from q. Because an individual sample of $\mathbf{z}$ is not produced by a function, but rather by a sampling process whose output changes every time we query it, it cannot be learned by back-propagation because the stochastic portion of the computation is not differentiable.

The solution, called the "reparametrization trick", is to first sample $\varepsilon \sim \mathcal{N}_k(0, I)$ and then compute $\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x})^{1/2}\varepsilon$. Thus, the ELBO would be written as

$$\begin{aligned} ELBO(q(\mathbf{z}|\mathbf{x}), p(\mathbf{x}|\mathbf{z})) =& \mathbb{E}_{\varepsilon\sim\mathcal{N}_k(0,I)}\left[\log p(\mathbf{x}|\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x})^{1/2}\varepsilon)\right] \\ &- D_{\mathrm{KL}}(q(\mathbf{z})||p(\mathbf{z})) \end{aligned} \tag{4.11}$$

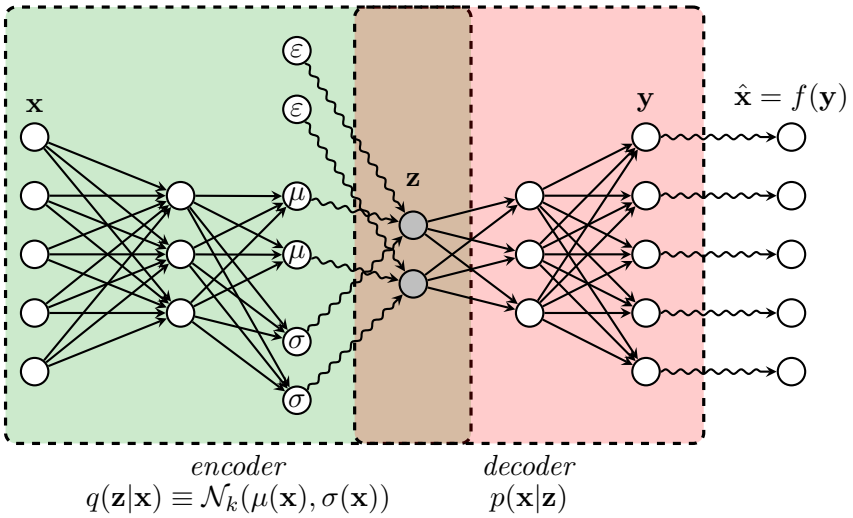

Figure 4.15: Diagram of a shallow VAE, note that we sample ε as an input layer, and we compute $\mathbf{z} = \mu(\mathbf{x}) + \sigma^{1/2}(\mathbf{x})\varepsilon$. This sampling allow us to back propagate the error and let us use stochastic gradient descent to train the autoencoder.

Note that the expectation in Equation (4.11) is with respect a random variable whose distribution is not a function of any of the variables whose derivatives we want to calculate. We are now able to back-propagate through the sampling operation, by regarding it as a deterministic operation with an extra input ε. This is shown schematically in Figure 4.15. In the particular case of a Bernoulli likelihood, the optimization problem to solve is

$$\begin{aligned}(d^\star, \mu^\star, \sigma^\star) \in \underset{(d,\mu,\sigma)\in\mathcal{D}\times\mathcal{M}\times\mathcal{S}}{\arg\max} \; & \mathbb{E}_{\varepsilon\sim\mathcal{N}_k(0,I)}\Bigg[\\ & \sum_{j=1}^{p} x_j \log(d_j(\mathbf{z})) + (1-x_j)\log(1-d_j(\mathbf{z}))\Big|\mathbf{z} = \mu(\mathbf{x}) + \sigma(\mathbf{x})^{1/2}\varepsilon\Bigg] \\ & - \frac{1}{2}\sum_{\kappa=1}^{k}\left[\sigma_\kappa(\mathbf{x}) + \mu_\kappa^2(\mathbf{x}) - \log\sigma_\kappa(\mathbf{x})\right]\end{aligned} \tag{4.12}$$

4.4.2 Example (MNIST Data Set, Continuation)

For the MNIST data we build a VAE with a similar architecture to our previous autoencoder, however some adaptations are necessary due to the sampling step in the latent space. The last layer of $\sigma(\mathbf{x})$ has an exponential activation function, while the last layer for $\mu(\mathbf{x})$ has a linear activation function. Then we compute $\mathbf{z} = \mu(\mathbf{x}) + \sigma^{1/2}(\mathbf{x})\varepsilon$, where $\varepsilon \sim \mathcal{N}_k(0, I)$.

We have taken $p(\mathbf{z}) \equiv \mathcal{N}_k(0, I)$, thus the optimization problem induced by the variational inference is expressed in Equation (4.12). As we have done before, we vary the dimension of the latent space to be $k = 2$ or $k = 98$. Figure 4.16 shows a diagram for this VAE.

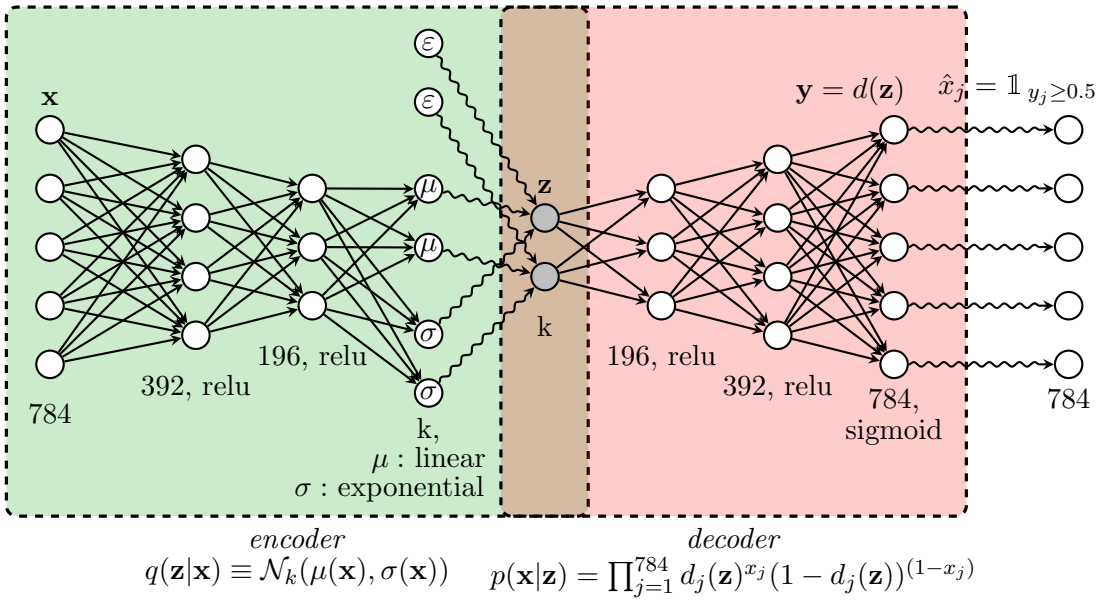

Figure 4.16: Deep VAE used in our experiments, we vary the dimension of the latent space k to be 2 or 98.

Figures 4.17 and 4.18 show the reconstruction of the first 10 images of the testing data set made by our VAE when the dimension of the latent space is $k = 2$ and $k = 98$, respectively. In Figures 4.19 and 4.20 we compare the latent space induced by the previous autoencoder built in Section 4.1.1 when $k = 2$ and the one induced by our VAE when $k = 2$. We can notice the regularity in the latent space obtained by the VAE. It is evident that there are large discontinuities in the space presented in Figure 4.19, these sparse regions may not correspond to meaningful points. On the other hand, the regularization term in the VAE encourages the training points to be (roughly) distributed as a Gaussian distribution, and there are far fewer discontinuities in Figure 4.20.

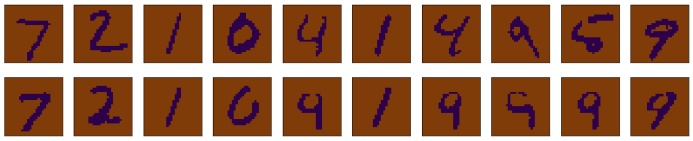

Figure 4.17: Reconstruction of the first 10 images of the testing MNIST data set using our VAE when the dimension of the latent space is $k = 2$.

4.5 Variational Autoencoders with Missing Values

We can use variational autoencoders to reconstruct data points with missing values in the same way as we did with denoising autoencoders in Section 4.2, just adding a corruption process $p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})$ that represents the conditional distribution over corrupted samples given the original data sample and the missing pattern, previous to the input of the autoencoder. However, since we are introducing a probability model based on

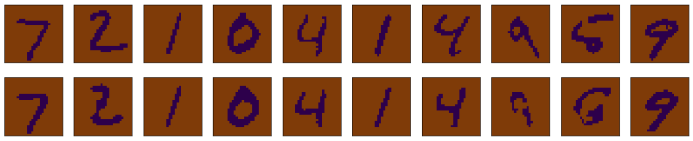

Figure 4.18: Reconstruction of the first 10 images of the testing MNIST data set using our VAE when the dimension of the latent space is $k = 98$.

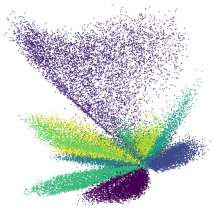

Figure 4.19: Latent space of the MNIST data set induced by our autoencoder when the dimension of the latent space is $k = 2$.

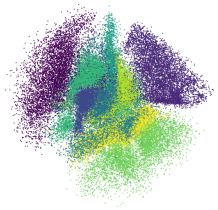

Figure 4.20: Latent space of the MNIST data set induced by our VAE when the dimension of the latent space is $k = 2$.

variational inference, it is interesting to analyze the induced optimization problem to solve.

Assume that $\tilde{\mathbf{x}}$ is a corrupted version of a data point $\mathbf{x}$, built with the observed part of $\mathbf{x}$ and its missingness pattern $\mathbf{m}$ in a similar way as we did in Equation (4.1). Let $p(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{z}, \mathbf{m})$ be the joint distribution of $\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{z}$ and $\mathbf{m}$. Conditioning in different order over the variables, we get two equivalent expressions for this distribution, given by

$$p(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{z}, \mathbf{m}) = p(\mathbf{z}|\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{m})p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})p(\mathbf{x}, \mathbf{m}) \tag{4.13}$$

$$p(\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{z}, \mathbf{m}) = p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{z}, \mathbf{m})p(\mathbf{x}, \mathbf{m}|\mathbf{z})p(\mathbf{z}) \tag{4.14}$$

On the other hand, since $\mathbf{z}$ is the reconstruction made from the corrupted observation $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}$ is defined through $\mathbf{x}$ and $\mathbf{m}$, it is easy to see that

$$p(\mathbf{z}|\mathbf{x}, \tilde{\mathbf{x}}, \mathbf{m}) = p(\mathbf{z}|\tilde{\mathbf{x}}) \tag{4.15}$$

and

$$p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{z}, \mathbf{m}) = p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m}) \tag{4.16}$$

Substituting Equations (4.15) and (4.16) in Equations (4.13) and (4.14) and matching the right hand sides, we get

$$p(\mathbf{z}|\tilde{\mathbf{x}})\cancel{p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})}p(\mathbf{x}, \mathbf{m}) = \cancel{p(\tilde{\mathbf{x}}|\mathbf{x}, \mathbf{m})}p(\mathbf{x}, \mathbf{m}|\mathbf{z})p(\mathbf{z}) \tag{4.17}$$

It is straightforward from Equation (4.17) that

$$p(\mathbf{z}|\tilde{\mathbf{x}}) = \frac{p(\mathbf{x}, \mathbf{m}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x}, \mathbf{m})} \tag{4.18}$$

Let us define the evidence and missingness mechanism lower bound (EMMELBO) as

$$\text{EMMELBO}(q(\mathbf{z}|\tilde{\mathbf{x}}), p(\mathbf{x}, \mathbf{m}|\mathbf{z})) = \log p(\mathbf{x}) + \log p(\mathbf{m}|\mathbf{x}) - D_{KL}(q(\mathbf{z}|\tilde{\mathbf{x}})||p(\mathbf{z}|\tilde{\mathbf{x}})) \quad (4.19)$$

where $(q(\mathbf{z}|\tilde{\mathbf{x}}), p(\mathbf{x}, \mathbf{m}|\mathbf{z})) \in \mathcal{Q} \times \mathcal{P}$, and $\mathcal{Q}$ and $\mathcal{P}$ are parametric families of distributions.

Because the difference between $\log p(\mathbf{x}) + \log p(\mathbf{m}|\mathbf{x})$ and $\text{EMMELBO}(q(\mathbf{z}|\tilde{\mathbf{x}}), p(\mathbf{x}, \mathbf{m}|\mathbf{z}))$ is given by a KL divergence which is always non-negative, we can see that $\text{EMMELBO}(q(\mathbf{z}|\tilde{\mathbf{x}}), p(\mathbf{x}, \mathbf{m}|\mathbf{z}))$ is a lower bound of $\log p(\mathbf{x}) + \log p(\mathbf{m}|\mathbf{x})$. Note that the EMMELBO is maximized when $q(\mathbf{z}|\tilde{\mathbf{x}})$ is the same distribution that $p(\mathbf{z}|\tilde{\mathbf{x}})$. From Equation (4.18), we can compute $D_{KL}(q(\mathbf{z}|\tilde{\mathbf{x}})||p(\mathbf{z}|\tilde{\mathbf{x}}))$ as

$$\begin{aligned} &D_{KL}(q(\mathbf{z}|\tilde{\mathbf{x}})||p(\mathbf{z}|\tilde{\mathbf{x}})) \\ &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[\log q(\mathbf{z}|\tilde{\mathbf{x}}) - \log p(\mathbf{z}|\tilde{\mathbf{x}})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}[\log q(\mathbf{z}|\tilde{\mathbf{x}}) - \log p(\mathbf{z}) - \log p(\mathbf{x}, \mathbf{m}|\mathbf{z}) + \log p(\mathbf{x}, \mathbf{m})] \end{aligned} \quad (4.20)$$

Substituting Equation (4.20) in Equation (4.19), we get the more convenient way to write the EMMELBO

$$\begin{aligned} &\text{EMMELBO}(q(\mathbf{z}|\tilde{\mathbf{x}}), p(\mathbf{x}, \mathbf{m}|\mathbf{z})) \\ &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[\log p(\mathbf{x}, \mathbf{m}|\mathbf{z})\right] - \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[\log q(\mathbf{z}|\tilde{\mathbf{x}}) - \log p(\mathbf{z})\right] \\ &= \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[\log p(\mathbf{x}, \mathbf{m}|\mathbf{z})\right] - D_{KL}(q(\mathbf{z}|\tilde{\mathbf{x}})||p(\mathbf{z})) \end{aligned} \quad (4.21)$$

From Equation (4.21) we can see that $q(\mathbf{z}|\tilde{\mathbf{x}})$ acts as the encoder and $p(\mathbf{x}, \mathbf{m}|\mathbf{z})$ as the decoder of a variational autoencoder that tries to maximize the EMMELBO. Notice that $p(\mathbf{x}, \mathbf{m}|\mathbf{z})$ decodes $\mathbf{z}$ into the original point $\mathbf{x}$ and the missingness pattern $\mathbf{m}$. Hence, we can model the decoded variables $\mathbf{y} = d(\mathbf{z})$ and $\mathbf{w} = g(\mathbf{z})$ with two neural networks that share part of their architecture and weights.

This observation has important repercussions when the autoencoder is used for generative purposes. In most of the applications we do not have access to the mechanism of missingness which can be difficult to capture with a simple model. An autoencoder that has been trained to maximize the EMMELBO might create a data point $\mathbf{x}_{new}$ and a missing patter $\mathbf{m}_{new}$ such that, if we delete the values of $\mathbf{x}_{new}$ in the places where $\mathbf{m}_{new} = 1$ the resulting observation would be a data point with missing values obtained with the same mechanism of missingness as the observations that we have in the original data set.

Creating new data points with the same mechanism of missingness might be important in applications where we would like to share the original data set with a third-party, but the data is susceptible to confidential information, hence we might create a new data set with the same data-missing mechanism than the original one that we could share. In many other applications, however, we are not interested in reconstructing the missing pattern, in this case we can set the decoder to be $p(\mathbf{x}|\mathbf{z})$ as in usual variational autoencoders, and instead of maximizing the EMMELBO, we could maximize

$$\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[\log p(\mathbf{x}|\mathbf{z})\right] - D_{KL}(q(\mathbf{z}|\tilde{\mathbf{x}})||p(\mathbf{z})) \quad (4.22)$$

As usual, with variational autoencoders we can set $q(\mathbf{z}|\tilde{\mathbf{x}}) \equiv \mathcal{N}_k(\mu(\tilde{\mathbf{x}}), \sigma(\tilde{\mathbf{x}}))$ and $p(\mathbf{z}) \equiv \mathcal{N}_k(0, I)$. A variational autoencoder that maximizes the EMMELBO can be trained using back-propagation and the reparametrization trick. Figure 4.21 shows a diagram of a VAE trained to maximize the EMMELBO.

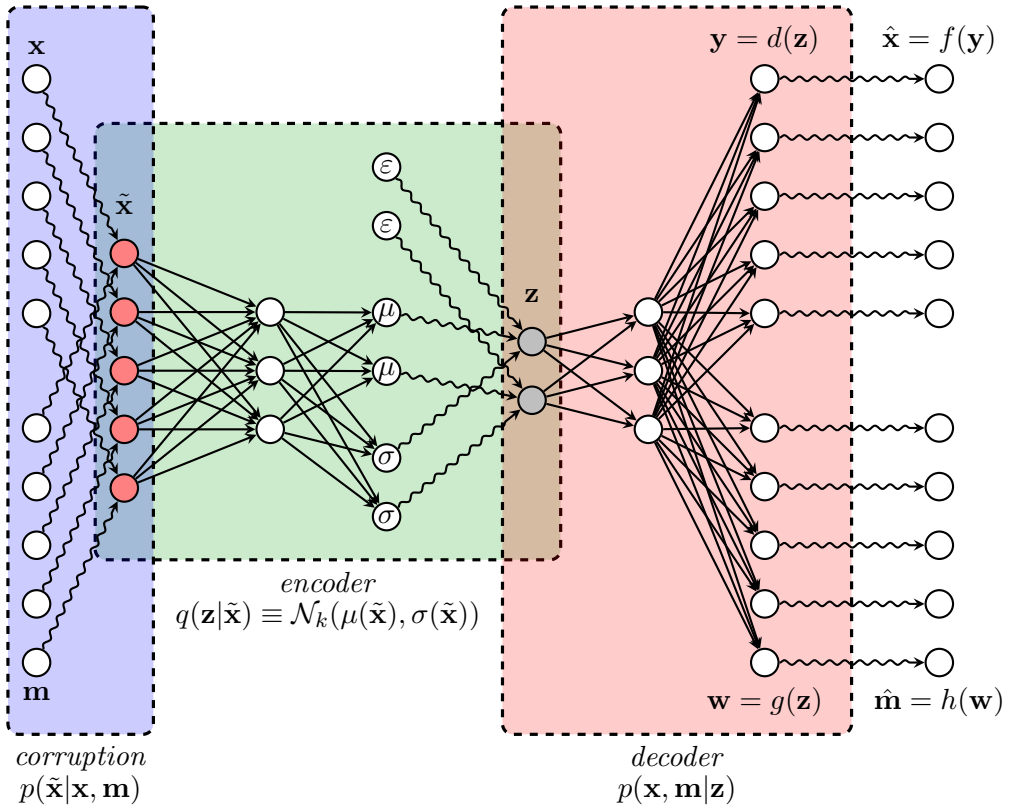

Figure 4.21: Diagram of a shallow VAE trained to maximize the EMMELBO, note that we sample ε as an input layer, and we compute $\mathbf{z} = \mu(\tilde{\mathbf{x}}) + \sigma^{1/2}(\tilde{\mathbf{x}})\varepsilon$. This sampling allows us to back-propagate the error so we can use stochastic gradient descent to train the autoencoder.

4.5.1 Example (MNIST Data Set, Continuation)

Consider once again the MNIST data and the data-missing mechanism that we introduced in Section 4.2.1 that creates $\tilde{\mathbf{x}}$ from the original point $\mathbf{x}$ and the missing pattern $\mathbf{m}$. Since we are only interested in the reconstruction of the original images, we construct a VAE that maximizes the criterion presented in Equation (4.22), instead of maximizing the EMMELBO. We have set $q(\mathbf{z}|\tilde{\mathbf{x}}) \equiv \mathcal{N}_k(\mu(\tilde{\mathbf{x}}), \sigma(\tilde{\mathbf{x}}))$ and $p(\mathbf{z}) \equiv \mathcal{N}_k(0, I)$. In Equation (4.10) we show that the optimization problem to solve with these assumptions is given by

$$(d^\star, \mu^\star, \sigma^\star) \in \underset{(d,\mu,\sigma)\in\mathcal{D}\times\mathcal{M}\times\mathcal{S}}{\arg\max} \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})} \left[\sum_{j=1}^{p} \tilde{x}_j \log(d_j(\mathbf{z})) + (1-\tilde{x}_j)\log(1-d_j(\mathbf{z})) \right]$$
$$- \frac{1}{2} \sum_{\kappa=1}^{k} \left[\sigma_\kappa(\tilde{\mathbf{x}}) + \mu_\kappa^2(\tilde{\mathbf{x}}) - \log \sigma_\kappa(\tilde{\mathbf{x}}) \right]$$

We have used the same encoder-decoder architecture as in our traditional autoencoder, which is represented schematically in Figure 4.16, except that the input now is $\tilde{\mathbf{x}}$ instead of $\mathbf{x}$. As it is usual now in our experiments, we vary the dimension of the latent space to be $k = 2$ or $k = 98$. As in our experiments with the denoising autoencoders for

missing data, we vary the number of missing pixels to be 157 or 704. In Figures 4.22 to 4.25 we show the results of the reconstructions. The original image is shown at the top, the image with missing values is shown in the middle and the reconstruction done by the VAE is shown at the bottom.

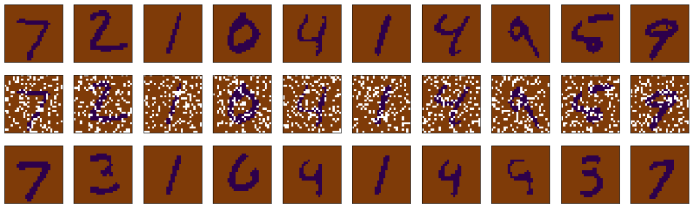

Figure 4.22: Reconstruction of the first 10 images of the testing MNIST data set, using our VAE for missing data when the dimension of the latent space is $k = 2$ and 157 pixels have missing values.

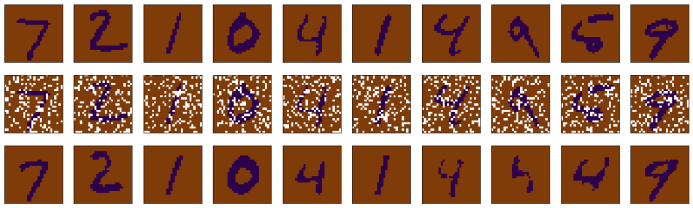

Figure 4.23: Reconstruction of the first 10 images of the testing MNIST data set, using our VAE for missing data when the dimension of the latent space is $k = 98$ and 157 pixels have missing values.

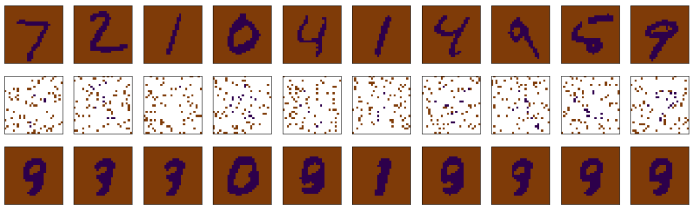

Figure 4.24: Reconstruction of the first 10 images of the testing MNIST data set, using our VAE for missing data when the dimension of the latent space is $k = 2$ and 704 pixels have missing values.

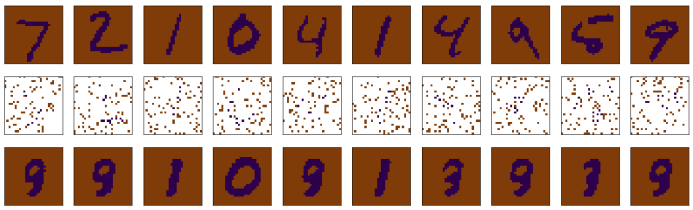

Figure 4.25: Reconstruction of the first 10 images of the testing MNIST data set, using our VAE for missing data when the dimension of the latent space is $k = 98$ and 704 pixels have missing values.

4.5.2 Extensions of Variational Autoencoders

Remember that traditional autoencoders maximize some function $L(\mathbf{x}, d(\mathbf{z}))$, where $\mathbf{z} = e(\mathbf{x})$ is the code of $\mathbf{x}$ and $L(\mathbf{x}, d(\mathbf{z}))$ penalizes the dissimilarity between $d(\mathbf{z})$ and $\mathbf{x}$, a usual choice for L is the log-likelihood $p(\mathbf{x}|\mathbf{z})$ where $d(\mathbf{z})$ is a parameter of this distribution, to be learned. Hence, training an autoencoder is equivalent to solve the optimization problem

$$(e^\star, d^\star) \in \underset{(e,d)\in\mathcal{E}\times\mathcal{D}}{\arg\max}\, \mathbb{E}_{\mathbf{z}\sim\delta_{e(\mathbf{x})}}\left[L(\mathbf{x}, d(\mathbf{z}))\right] \tag{4.23}$$

On the other hand, training a traditional VAE is equivalent to solve the optimization problem

$$\begin{aligned}(q^\star(\mathbf{z}|\mathbf{x}), p^\star(\mathbf{x}|\mathbf{z})) \in \underset{(q(\mathbf{z}|\mathbf{x}),p(\mathbf{x}|\mathbf{z}))\in\mathcal{Q}\times\mathcal{P}}{\arg\max}\ & \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[L(\mathbf{x}, d(\mathbf{z}))\right] \\ & - \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[L(q(\mathbf{z}|\mathbf{x}), p(\mathbf{z}))\right]\end{aligned} \tag{4.24}$$

where $\mathcal{Q}$ and $\mathcal{P}$ are parametric families of distributions, $L(\mathbf{x}, d(\mathbf{z})) = \log p(\mathbf{x}|\mathbf{z})$, $d(\mathbf{z})$ and $L(q(\mathbf{z}|\mathbf{x}), p(\mathbf{z})) = \log q(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z})$.

We could add a regularization parameter $\beta > 0$ into Equation (4.24), changing the optimization problem to be

$$\begin{aligned}(q^\star(\mathbf{z}|\mathbf{x}), p^\star(\mathbf{x}|\mathbf{z})) \in \underset{(q(\mathbf{z}|\mathbf{x}),p(\mathbf{x}|\mathbf{z}))\in\mathcal{Q}\times\mathcal{P}}{\arg\max}\ & \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[L(\mathbf{x}, d(\mathbf{z}))\right] \\ & - \beta\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\mathbf{x})}\left[L(q(\mathbf{z}|\mathbf{x}), p(\mathbf{z}))\right]\end{aligned} \tag{4.25}$$

Small values of β would make the optimization problem presented in Equation (4.25) to be more similar to the optimization problem in Equation (4.23), and hence will favor exact reconstruction at a price of a less regular latent space. On the other hand, if β is chosen to be extremely large, then all points will have the same hidden distribution $p(\mathbf{z})$.

Additionally to the regularization parameter β, we could change the functions $L(\mathbf{x}, d(\mathbf{z}))$ and $L(q(\mathbf{z}|\mathbf{x}), p(\mathbf{z}))$ to other functions that make sense into the particular problem that we have on hand and to the final use of the variational autoencoder. However, the new criterion could not have the same interpretation as a lower bound of the evidence.

In Sections 4.2 and 4.5 we explain how denoising autoencoders and variational autoencoders could be used to handle points with missing values. Furthermore, we discuss

how variational autoencoders could produce not only new points, but also new missing patterns such that they are created with the same mechanism of missingness that the original observations. Thus, we can add an output $g(\mathbf{z})$ that tries to reconstruct the missing pattern, where $g \in \mathcal{G}$ and $\mathcal{G}$ is a family of parametric functions. This output could be added in the same way to denoising autoencoders. Therefore, when dealing with missing values, we could solve the optimization problems

$$(e^\star, d^\star) \in \underset{(e,d)\in\mathcal{E}\times\mathcal{D}}{\arg\max}\, \mathbb{E}_{\mathbf{z}\sim\delta_{e(\tilde{\mathbf{x}})}}\left[L(\tilde{\mathbf{x}}, d(\mathbf{z}))\right] + \beta_1 \mathbb{E}_{\mathbf{z}\sim\delta_{e(\tilde{\mathbf{x}})}}\left[L(\mathbf{m}, g(\mathbf{z}))\right] \tag{4.26}$$

when we train a denoising autoencoder, and

$$\begin{aligned}(q^\star(\mathbf{z}|\tilde{\mathbf{x}}), d^\star, g^\star) \in \underset{(q(\mathbf{z}|\mathbf{x}),d,g)\in\mathcal{Q}\times\mathcal{D}\times\mathcal{G}}{\arg\max}\; & \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[L(\tilde{\mathbf{x}}, d(\mathbf{z}))\right] \\ & + \beta_1 \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[L(\mathbf{m}, g(\mathbf{z}))\right] \\ & - \beta_2 \mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}})}\left[L(q(\mathbf{z}|\tilde{\mathbf{x}}), p(\mathbf{z}))\right]\end{aligned} \tag{4.27}$$

when we train a variational autoencoder, where $\beta_1, \beta_2 \geq 0$. In this case, large values of β_1 would encourage the autoencoders to learn the data-missing mechanism rather than reconstruct the original input and small values of β_2 will favor exact reconstruction at a price of a less regular latent space.

In the work of Im et al. (2017), they consider the use of VAEs with a denoising criterion. They part from the idea that denoising criteria helps to achieve good generalization since the low dimensional representation is robust to small (Vincent et al., 2008) noise. Ryu, Kim, and Kim (2020) used this denoising criterion in a study to impute missing values using autoencoders. This criterion differs from the EMMELBO in its form an conception. In the work of Im et al. (2017) they consider that we artificially inject noise to the original input through a corruption process that is completely determined by the user and then it is irrelevant to learn the mechanism that corrupts the original data. In our work, the EMMELBO is developed from the idea that the corruption process is through an unknown mechanism of missingness.

With similar ideas to the development of the EMMELBO, Collier, Nazabal, and Williams (2020) propose to maximize

$$\mathbb{E}_{\mathbf{z}\sim q(\mathbf{z}|\tilde{\mathbf{x}},\mathbf{m})}\left[\log p(\mathbf{x}_o|\mathbf{z},\mathbf{m})\right] - D_{KL}(q(\mathbf{z}|\tilde{\mathbf{x}},\mathbf{m})||p(\mathbf{z}))$$

where $\mathbf{x}_o$ is the observed part of $\mathbf{x}$. Note that in this criterion the missingness pattern $\mathbf{m}$ does not appear in the corruption process, instead it appears in $q(\mathbf{z}|\tilde{\mathbf{x}},\mathbf{m})$ and $p(\mathbf{x}_o|\mathbf{z},\mathbf{m})$ allowing the encoder and decoder networks to distinguish between observed elements of the input and missing values. Furthermore, this criterion does not require to know the original input $\mathbf{x}$.

Since different autoencoders maximize different functions, that are not necessarily in the same scale, we need some way to compare them. One function that we could use for this purpose is the area under the curve (AUC), that could be computed as

$$\text{AUC}(\mathbf{x},\mathbf{y}) = \frac{1}{n_+ n_-}\sum_{i<j} \mathbb{1}_{(x_i - x_j)(y_i - y_j)>0}$$

where $n_+ = \sum_{i=1}^p \mathbb{1}_{x_i=1}$ and $n_- = \sum_{i=1}^p \mathbb{1}_{x_i=0}$. Note that this function is maximized if for all $x_i = 1$ and for all $x_j = 0$, we have $y_i > y_j$.

In some applications, like recommender systems we do not always have access to the original point $\mathbf{x}$ (it is difficult that one user has interacted with all the items or that

one item has been evaluated by all users), instead we only have access to its incomplete counterpart $\tilde{\mathbf{x}}$, so we cannot compare its reconstructed version $\mathbf{y}$ to the original $\mathbf{x}$. Luckily, one advantage of the AUC is that it can be calculated just in those parts of $\mathbf{x}$ that are observed. If we define $\tilde{\mathbf{x}}$ according to the corruption process defined in Section 4.2.1, then the AUC can be computed as

$$\text{AUC}(\tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{\tilde{n}_+ \tilde{n}_-} \sum_{\substack{\tilde{x}_i, \tilde{x}_j \in \{0,1\} \\ i<j}} \mathbb{1}_{(\tilde{x}_i - \tilde{x}_j)(y_i - y_i) > 0}$$

where $\tilde{n}_+ = \sum_{i=}^{p} \mathbb{1}_{\tilde{x}_i=1}$ and $\tilde{n}_- = \sum_{i=1}^{p} \mathbb{1}_{\tilde{x}_i=0}$

If we decide to compare the autoencoders using the AUC, we could try to train them to maximize this function, that is, we could take $L(\tilde{\mathbf{x}}, \mathbf{y}) = \text{AUC}(\tilde{\mathbf{x}}, \mathbf{y})$ in Equations (4.26) and (4.27). However, the AUC is not a differentiable function. So, we cannot apply back-propagation to train an autoencoder that tries to maximize the AUC. Hence, we can use the approximated function suggested in Vogel, Bellet, and Clémençon (2020), given by

$$L(\tilde{\mathbf{x}}, \mathbf{y}) = \frac{1}{\tilde{n}_+ \tilde{n}_-} \sum_{\substack{\tilde{x}_i, \tilde{x}_j \in \{0,1\} \\ i<j}} \sigma((\tilde{x}_i - \tilde{x}_j)(y_i - y_j)) \tag{4.28}$$

where $\sigma(x) = \frac{1}{1+e^{-x}}$

4.6 Recommender Systems with Autoencoders and Future Work

Deep learning techniques (and particularly autoencoders) are becoming the dominant base for recommender systems (Dacrema, Cremonesi, and Jannach, 2019; Bacuet, 2019). Neural network-based collaborative filtering (He et al., 2017) generalizes Matrix Factorization by replacing the inner product with a neural network that can learn an arbitrary function from the data. Wang, Wang, and Yeung (2015) propose the Collaborative Deep Learning (CDL) method which is based on Stacked Denoising Autoencoders (SDAE) and Collaborative Filtering, similarly Li, Kawale, and Fu (2015) consider marginalized deniosing autoencoders (a variant of SDAE), these works aim to find a deep latent representation for content information. Li and She (2017) propose the use of a Collaborative Variational Autoencoder which learns the distribution for content in the latent space instead of the observation space and also learns implicit relationships between items and users from both content and rating. Liang et al. (2018) also make use of Variational Autoencoders considering the multinomial log-likelihood and the loss function presented in Equation (4.25) where β is selected through a simple heuristic in which they start with $\beta = 0$ and gradually increase $\beta = 1$, selecting the value associated with the best results.

Some of the ideas developed in this chapter could also be applied, for example we could built an autoencoder which considers the loss function given by the approximated AUC presented in Equation (4.28). Such autoencoder would learn to put a higher value on those items that the user might like while assigning lower values to those items that he/she might not like.

References

Aggarwal, Charu C et al. (2018). *Neural networks and deep learning*. Springer (see pp. 15, 17).

Aghdam, Hamed Habibi and Elnaz Jahani Heravi (2017). "Guide to convolutional neural networks". In: *New York, NY: Springer* 10, pp. 978–973 (see p. 15).

Alain, Guillaume and Yoshua Bengio (2014). "What regularized auto-encoders learn from the data-generating distribution". In: *The Journal of Machine Learning Research* 15.1, pp. 3563–3593 (see p. 77).

Azur, Melissa J et al. (2011). "Multiple imputation by chained equations: what is it and how does it work?" In: *International journal of methods in psychiatric research* 20.1, pp. 40–49 (see p. 76).

Bacuet, Quentin (2019). *How Variational Autoencoders make classical recommender systems obsolete*. URL: https://medium.com/snipfeed/how-variational-autoencoders-make-classical-recommender-systems-obsolete-4df8bae51546 (see p. 91).

Barber, David (2012). *Bayesian reasoning and machine learning*. Cambridge University Press (see p. 79).

Barber, David and P de van Laar (1999). "Variational cumulant expansions for intractable distributions". In: *Journal of Artificial Intelligence Research* 10, pp. 435–455 (see p. 79).

Barcenas, Roberto (2020). "Local sampling SVM and minority oversampling boosting: sampling-based approaches for statistical classification". In: *Ph. D. dissertation* (see p. 5).

Beaulieu-Jones, Brett K and Jason H Moore (2017). "Missing data imputation in the electronic health record using deeply learned autoencoders". In: *Pacific Symposium on Biocomputing 2017*. World Scientific, pp. 207–218 (see p. 76).

Bengio, Yoshua et al. (2013). "Generalized denoising auto-encoders as generative models". In: *Advances in neural information processing systems* 26, pp. 899–907 (see pp. 76, 77).

Biau, Gérard (2012). "Analysis of a random forests model". In: *Journal of Machine Learning Research* 13.Apr, pp. 1063–1095 (see pp. 10, 51).

Biau, Gérard, Luc Devroye, and Gábor Lugosi (2008). "Consistency of random forests and other averaging classifiers". In: *Journal of Machine Learning Research* 9.Sep, pp. 2015–2033 (see pp. 10, 51).

Blei, David M, Michael I Jordan, et al. (2006). "Variational inference for Dirichlet process mixtures". In: *Bayesian analysis* 1.1, pp. 121–143 (see p. 79).

Blei, David M, Alp Kucukelbir, and Jon D McAuliffe (2017). "Variational inference: A review for statisticians". In: *Journal of the American statistical Association* 112.518, pp. 859–877 (see p. 79).

Bourlard, Hervé and Yves Kamp (1988). "Auto-association by multilayer perceptrons and singular value decomposition". In: *Biological cybernetics* 59.4-5, pp. 291–294 (see p. 17).

Braun, Michael and Jon McAuliffe (2010). "Variational inference for large-scale models of discrete choice". In: *Journal of the American Statistical Association* 105.489, pp. 324–335 (see p. 79).

Breiman, Leo (1996). "Bagging predictors". In: *Machine learning* 24.2, pp. 123–140 (see pp. 9, 29).

— (2001). "Random forests". In: *Machine learning* 45.1, pp. 5–32 (see pp. 8, 11, 40).

— (2003). *Setting up, using, and understanding random forests V4.0*. URL: https://www.stat.berkeley.edu/~breiman/Using_random_forests_v4.0.pdf (see pp. 19, 27, 28, 35, 40).

— (2004). "Consistency for a simple model of random forests". In: (see pp. 10, 51).

Breiman, Leo et al. (1984). *Classification and regression trees*. Chapman and Hall/CRC (see pp. 11, 21, 28).

Chen, Tianqi and Carlos Guestrin (2016). "Xgboost: A scalable tree boosting system". In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794 (see p. 29).

Collier, Mark, Alfredo Nazabal, and Christopher KI Williams (2020). "VAEs in the Presence of Missing Data". In: *arXiv preprint arXiv:2006.05301* (see p. 90).

Costa, Adriana Fonseca et al. (2018). "Missing data imputation via denoising autoencoders: the untold story". In: *International Symposium on Intelligent Data Analysis*. Springer, pp. 87–98 (see pp. 67, 72, 73, 76).

Creswell, Antonia et al. (2018). "Generative adversarial networks: An overview". In: *IEEE Signal Processing Magazine* 35.1, pp. 53–65 (see p. 17).

Crookston, Nicholas L and Andrew O Finley (2008). "yaImpute: an R package for kNN imputation". In: *Journal of Statistical Software. 23 (10). 16 p.* (see p. 28).

Cutler, Adele and Guohua Zhao (2001). "Pert-perfect random tree ensembles". In: *Computing Science and Statistics* 33, pp. 490–497 (see p. 51).

Dacrema, Maurizio Ferrari, Paolo Cremonesi, and Dietmar Jannach (2019). "Are we really making much progress? A worrying analysis of recent neural recommendation approaches". In: *Proceedings of the 13th ACM Conference on Recommender Systems*, pp. 101–109 (see p. 91).

Dempster, Arthur P, Nan M Laird, and Donald B Rubin (1977). "Maximum likelihood from incomplete data via the EM algorithm". In: *Journal of the Royal Statistical Society: Series B (Methodological)* 39.1, pp. 1–22 (see pp. 29, 76, 79).

Deng, Li and Yang Liu (2018). *Deep learning in natural language processing*. Springer (see p. 15).

Denil, Misha, David Matheson, and Nando Freitas (2013). "Consistency of online random forests". In: *International conference on machine learning*, pp. 1256–1264 (see p. 52).

Devroye, Luc and Gábor Lugosi (2012). *Combinatorial methods in density estimation*. Springer Science & Business Media (see p. 8).

Doersch, Carl (2016). "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908* (see pp. 17, 81).

Dua, Dheeru and Casey Graff (2017). *UCI Machine Learning Repository*. URL: http://archive.ics.uci.edu/ml (see p. 28).

Efromovich, Sam (2008). *Nonparametric curve estimation: methods, theory, and applications*. Springer Science & Business Media (see p. 8).

Erhan, Dumitru et al. (2010). "Why does unsupervised pre-training help deep learning?" In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 201–208 (see p. 76).

Farhangfar, Alireza, Lukasz Kurgan, and Jennifer Dy (2008). "Impact of imputation of missing values on classification error for discrete data". In: *Pattern Recognition* 41.12, pp. 3692–3705 (see pp. 21, 28).

Feelders, Ad (1999). "Handling missing data in trees: surrogate splits or statistical imputation?" In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer, pp. 329–334 (see pp. 21, 27, 28).

Friedman, Jerome H et al. (1991). "Multivariate adaptive regression splines". In: *The annals of statistics* 19.1, pp. 1–67 (see p. 29).

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2009). *The elements of statistical learning*. 2nd ed. Springer series in statistics New York (see p. 21).

García-Laencina, Pedro J, José-Luis Sancho-Gómez, and AníBal R Figueiras-Vidal (2013). "Classifying patterns with missing values using multi-task learning perceptrons". In: *Expert Systems with Applications* 40.4, pp. 1333–1341 (see p. 76).

Genuer, Robin (2012). "Variance reduction in purely random forests". In: *Journal of Nonparametric Statistics* 24.3, pp. 543–562 (see pp. 10, 51).

Gini, Corrado (1912). "Variabilità e mutabilità". In: *Reprinted in Memorie di metodologica statistica (Ed. Pizetti E, Salvemini, T). Rome: Libreria Eredi Virgilio Veschi* (see p. 11).

Glorot, Xavier, Antoine Bordes, and Yoshua Bengio (2011). "Domain adaptation for large-scale sentiment classification: A deep learning approach". In: *ICML* (see p. 68).

Gondara, Lovedeep and Ke Wang (2017). "Recovering loss to followup information using denoising autoencoders". In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 1936–1945 (see p. 76).

— (2018). "Mida: Multiple imputation using denoising autoencoders". In: *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, pp. 260–272 (see pp. 72, 76).

Goodfellow, Ian et al. (2014). "Generative adversarial nets". In: *Advances in neural information processing systems*, pp. 2672–2680 (see p. 17).

Goodfellow, Ian et al. (2016). *Deep learning*. Vol. 1. 2. MIT press Cambridge (see pp. 14, 15, 68, 69).

Goyal, Palash, Sumit Pandey, and Karan Jain (2018). "Deep learning for natural language processing". In: *Deep Learning for Natural Language Processing: Creating Neural Networks with Python [Berkeley, CA]: Apress*, pp. 138–143 (see p. 14).

Györfi, László et al. (2002). *A distribution-free theory of nonparametric regression*. Springer Science & Business Media (see p. 8).

Hapfelmeier, Alexander, Torsten Hothorn, and Kurt Ulm (2012). "Recursive partitioning on incomplete data using surrogate decisions and multiple imputation". In: *Computational Statistics & Data Analysis* 56.6, pp. 1552–1565 (see pp. 21, 28, 31, 48).

He, Xiangnan et al. (2017). "Neural collaborative filtering". In: *Proceedings of the 26th international conference on world wide web*, pp. 173–182 (see p. 91).

Hinton, Geoffrey E and Richard Zemel (1993). "Autoencoders, minimum description length and Helmholtz free energy". In: *Advances in neural information processing systems* 6, pp. 3–10 (see p. 17).

Hothorn, Torsten, Kurt Hornik, and Achim Zeileis (2006). "Unbiased recursive partitioning: A conditional inference framework". In: *Journal of Computational and Graphical statistics* 15.3, pp. 651–674 (see pp. 12, 28, 48).

Hothorn, Torsten and Achim Zeileis (2015). "partykit: A Modular Toolkit for Recursive Partytioning in R". In: *Journal of Machine Learning Research* 16.118, pp. 3905–3909. URL: http://jmlr.org/papers/v16/hothorn15a.html (see p. 22).

Im, Daniel J. et al. (2017). "Denoising criterion for variational auto-encoding framework". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1 (see p. 90).

Ishioka, Tsunenori (2013). "Imputation of missing values for unsupervised data using the proximity in random forests". In: *International Conference on Mobile, Hybrid, and On-line Learning. Nice*, pp. 30–36 (see pp. 20, 27, 28, 35, 48).

Ishwaran, Hemant and Udaya B Kogalur (2010). "Consistency of random survival forests". In: *Statistics & probability letters* 80.13-14, pp. 1056–1064 (see pp. 10, 52).

Jarrett, Kevin et al. (2009). "What is the best multi-stage architecture for object recognition?" In: *2009 IEEE 12th international conference on computer vision*. IEEE, pp. 2146–2153 (see p. 68).

Jordan, Michael I et al. (1999). "An introduction to variational methods for graphical models". In: *Machine learning* 37.2, pp. 183–233 (see p. 78).

Josse, Julie et al. (2019). "On the consistency of supervised learning with missing values". In: *arXiv preprint arXiv:1902.06931* (see pp. 22, 29, 48).

Kingma, Diederik P and Max Welling (2013). "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (see pp. 17, 77, 82).

— (2019). "An introduction to variational autoencoders". In: *arXiv preprint arXiv:1906.02691* (see p. 17).

Kucukelbir, Alp et al. (2017). "Automatic differentiation variational inference". In: *The Journal of Machine Learning Research* 18.1, pp. 430–474 (see p. 79).

Laurent, Beatrice and Pascal Massart (2000). "Adaptive estimation of a quadratic functional by model selection". In: *Annals of Statistics*, pp. 1302–1338 (see p. 59).

LeCun, Y. (1987). "Modeles connexionnistes de lapprentissage". In: *Ph. D. dissertation* (see p. 17).

Lee, Honglak, Chaitanya Ekanadham, and Andrew Ng (2007). "Sparse deep belief net model for visual area V2". In: *Advances in neural information processing systems* 20, pp. 873–880 (see p. 76).

Leisink, Martijn AR and Hilbert J Kappen (2001). "A tighter bound for graphical models". In: *Neural Computation* 13.9, pp. 2149–2171 (see p. 79).

Li, Sheng, Jaya Kawale, and Yun Fu (2015). "Deep collaborative filtering via marginalized denoising auto-encoder". In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 811–820 (see p. 91).

Li, Xiaopeng and James She (2017). "Collaborative variational autoencoder for recommender systems". In: *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 305–314 (see p. 91).

Liang, Dawen et al. (2018). "Variational autoencoders for collaborative filtering". In: *Proceedings of the 2018 World Wide Web Conference*, pp. 689–698 (see p. 91).

Lin, Yi and Yongho Jeon (2006). "Random forests and adaptive nearest neighbors". In: *Journal of the American Statistical Association* 101.474, pp. 578–590 (see p. 51).

Little, Roderick JA and Donald B Rubin (2002). *Statistical analysis with missing data*. 2nd ed. John Wiley & Sons (see p. 18).

Louppe, Gilles (2014). "Understanding random forests: From theory to practice". In: *arXiv preprint arXiv:1407.7502* (see p. 5).

Ma, Qian et al. (2020). "MIDIA: exploring denoising autoencoders for missing data imputation". In: *Data Mining and Knowledge Discovery* 34.6, pp. 1859–1897 (see pp. 72, 73, 75).

MacKay, David JC (1997). *Ensemble learning for hidden Markov models*. Tech. rep. (see p. 79).

Mallinson, H and A Gammerman (2003). "Imputation using support vector machines". In: *Department of Computer Science. Royal Holloway, University of London. Egham, UK* (see p. 76).

Meinshausen, Nicolai (2006). "Quantile regression forests". In: *Journal of Machine Learning Research* 7.Jun, pp. 983–999 (see p. 52).

Mentch, Lucas and Giles Hooker (2016). "Quantifying uncertainty in random forests via confidence intervals and hypothesis tests". In: *The Journal of Machine Learning Research* 17.1, pp. 841–881 (see p. 51).

Minka, Thomas P (2001). "Expectation propagation for approximate Bayesian inference". In: *Uncertainty in Artificial Inteligence* (see p. 79).

Minka, Tom et al. (2005). *Divergence measures and message passing*. Tech. rep. Technical report, Microsoft Research (see p. 79).

Mitchell, Lawrence et al. (2011). "A parallel random forest classifier for R". In: *Proceedings of the second international workshop on Emerging computational methods for the life sciences*, pp. 1–6 (see p. 5).

Mohri, Mehryar, Afshin Rostamizadeh, and Ameet Talwalkar (2018). *Foundations of machine learning*. MIT press (see p. 5).

Morgan, James N and Robert C Messenger (1973). "THAID, a sequential analysis program for the analysis of nominal scale dependent variables". In: (see p. 11).

Müller, Peter and Fernando A Quintana (2004). "Nonparametric Bayesian data analysis". In: *Statistical science*, pp. 95–110 (see p. 8).

Nair, Vinod and Geoffrey E Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *ICML* (see p. 68).

Ning, Xiuli et al. (2017). "Missing data of quality inspection imputation algorithm base on stacked denoising auto-encoder". In: *2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)*. IEEE, pp. 84–88 (see p. 76).

Oba, Shigeyuki et al. (2003). "A Bayesian missing value estimation method for gene expression profile data". In: *Bioinformatics* 19.16, pp. 2088–2096 (see p. 28).

Quinlan, J. Ross (1986). "Induction of decision trees". In: *Machine learning* 1.1, pp. 81–106 (see pp. 11, 21).

— (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. (see pp. 11, 21).

Ramachandran, Prajit, Barret Zoph, and Quoc V Le (2017). "Searching for activation functions". In: *arXiv preprint arXiv:1710.05941* (see p. 68).

Sparse feature learning for deep belief networks (2008), pp. 1185–1192 (see p. 76).

Rieger, Anna, Torsten Hothorn, and Carolin Strobl (2010). "Random forests with missing values in the covariates". In: (see pp. 3, 21, 27–32, 48).

Rifai, Salah et al. (2012). "A generative process for sampling contractive auto-encoders". In: *arXiv preprint arXiv:1206.6434* (see p. 77).

Rocca, Joseph (2019). *Understanding Variational Autoencoders (VAEs)*. URL: https://towardsdatascience.com/understanding-variational-autoencoders-vaes-f70510919f73 (see pp. 15, 17, 81).

Rubin, Donald B (1976). "Inference and missing data". In: *Biometrika* 63.3, pp. 581–592 (see p. 18).

Rubin, Donald B (1996). "Multiple imputation after 18+ years". In: *Journal of the American statistical Association* 91.434, pp. 473–489 (see p. 21).

— (2004). *Multiple imputation for nonresponse in surveys*. Vol. 81. John Wiley & Sons (see p. 21).

Ryu, Seunghyoung, Minsoo Kim, and Hongseok Kim (2020). "Denoising Autoencoder-Based Missing Value Imputation for Smart Meters". In: *IEEE Access* 8, pp. 40656–40666 (see pp. 72, 75, 90).

Schafer, Joseph L (1997). *Analysis of incomplete multivariate data*. CRC press (see p. 28).

Schafer, Joseph L and Maren K Olsen (1998). "Multiple imputation for multivariate missing-data problems: A data analyst's perspective". In: *Multivariate behavioral research* 33.4, pp. 545–571 (see p. 28).

Scornet, Erwan (2016). "On the asymptotics of random forests". In: *Journal of Multivariate Analysis* 146, pp. 72–83 (see p. 51).

Scornet, Erwan, Gérard Biau, and Jean-Philippe Vert (2015). "Consistency of random forests". In: *The Annals of Statistics* 43.4, pp. 1716–1741 (see pp. 3, 10, 51, 52).

Shalev-Shwartz, Shai and Shai Ben-David (2014). *Understanding machine learning: From theory to algorithms*. Cambridge university press (see p. 5).

Shannon, Claude Elwood (1948). "A mathematical theory of communication". In: *Bell system technical journal* 27.3, pp. 379–423 (see p. 11).

Skansi, Sandro (2018). *Introduction to Deep Learning: from logical calculus to artificial intelligence*. Springer (see p. 15).

Stekhoven, Daniel J and Peter Bühlmann (2011). "MissForest—non-parametric missing value imputation for mixed-type data". In: *Bioinformatics* 28.1, pp. 112–118 (see pp. 20, 27, 28, 35, 48).

Strasser, Helmut and Christian Weber (1999). "On the asymptotic theory of permutation statistics". In: (see p. 12).

Strobl, Carolin, Anne-Laure Boulesteix, and Thomas Augustin (2007). "Unbiased split selection for classification trees based on the Gini index". In: *Computational Statistics & Data Analysis* 52.1, pp. 483–501 (see p. 11).

Therneau, Terry M, Elizabeth J Atkinson, et al. (1997). *An introduction to recursive partitioning using the RPART routines*. Tech. rep. Technical report Mayo Foundation (see pp. 21, 28).

Thompson, Steven K. (2012). *Sampling*. 3rd ed. John Wiley & Sons (see p. 19).

Tolstikhin, Ilya et al. (2017). "Wasserstein auto-encoders". In: *arXiv preprint arXiv:1711.01558* (see p. 75).

Troyanskaya, Olga et al. (2001). "Missing value estimation methods for DNA microarrays". In: *Bioinformatics* 17.6, pp. 520–525 (see pp. 28, 48, 76).

Tsybakov, Alexandre B (2008). *Introduction to nonparametric estimation*. Springer Science & Business Media (see p. 8).

Twala, Beth, MC Jones, and David J Hand (2008). "Good methods for coping with missing data in decision trees". In: *Pattern Recognition Letters* 29.7, pp. 950–956 (see pp. 21, 27, 29, 37, 48).

Van Buuren, Stef et al. (2006). "Fully conditional specification in multivariate imputation". In: *Journal of statistical computation and simulation* 76.12, pp. 1049–1064 (see pp. 21, 28, 48, 76).

Venables and Ripley (2002). "Modern Applied Statistics with S". In: *Springer, New York* 1228, p. 1229 (see p. 21).

Vincent, Pascal (2011). "A connection between score matching and denoising autoencoders". In: *Neural computation* 23.7, pp. 1661–1674 (see p. 77).

Vincent, Pascal et al. (2008). "Extracting and composing robust features with denoising autoencoders". In: *Proceedings of the 25th international conference on Machine learning*, pp. 1096–1103 (see pp. 72, 76, 90).

Vincent, Pascal et al. (2010). "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion." In: *Journal of machine learning research* 11.12 (see p. 76).

Vogel, Robin, Aurélien Bellet, and Stéphan Clémençon (2020). "Learning Fair Scoring Functions: Fairness Definitions, Algorithms and Generalization Bounds for Bipartite Ranking". In: *arXiv preprint arXiv:2002.08159* (see p. 91).

Wager, Stefan and Susan Athey (2018). "Estimation and inference of heterogeneous treatment effects using random forests". In: *Journal of the American Statistical Association* 113.523, pp. 1228–1242 (see p. 51).

Wainwright, Martin J and Michael Irwin Jordan (2008). *Graphical models, exponential families, and variational inference*. Now Publishers Inc (see p. 79).

Wang, Hao, Naiyan Wang, and Dit-Yan Yeung (2015). "Collaborative deep learning for recommender systems". In: *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 1235–1244 (see p. 91).

Waterhouse, Steve R, David MacKay, and Anthony J Robinson (1996). "Bayesian methods for mixtures of experts". In: *Advances in neural information processing systems*, pp. 351–357 (see p. 79).

Xia, Jing et al. (2017). "Adjusted weight voting algorithm for random forests in handling missing values". In: *Pattern Recognition* 69, pp. 52–60 (see p. 76).

Xie, Junyuan, Linli Xu, and Enhong Chen (2012). "Image denoising and inpainting with deep neural networks". In: *Advances in neural information processing systems* 25, pp. 341–349 (see p. 76).

Yedidia, Jonathan S, William T Freeman, and Yair Weiss (2001). "Generalized belief propagation". In: *Advances in neural information processing systems*, pp. 689–695 (see p. 79).

Zhao, Guohua (2000). "A new perspective on classification". In: *Ph. D. dissertation* (see p. 51).

Zhu, Ruoqing, Donglin Zeng, and Michael R Kosorok (2015). "Reinforcement learning trees". In: *Journal of the American Statistical Association* 110.512, pp. 1770–1784 (see pp. 10, 52).

A

Decomposition of the CART Criterion

Remember that the modified CART criterion is defined as

$$\begin{aligned} L_n\left(A,d,w\right) =& \frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_A\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A} \\ &- \frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_{A_L}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z} \\ &- \frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_{A_R}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, z\leq\widehat{\mathbf{X}}^{(h)}_{i,out}\leq b^{(h)}} \end{aligned}$$

Note that

$$\begin{aligned} &\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_A\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A} \\ &= \frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_A\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\mathbf{M}^{(h)}_i=0} \\ &\quad + \frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_A\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\mathbf{M}^{(h)}_i=1} \\ &= \frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i+\widehat{Y}^{(h)}_{A,obs}-\widehat{Y}^{(h)}_{A,obs}-\widehat{Y}_A\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\mathbf{M}^{(h)}_i=0} \\ &\quad + \frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i+\widehat{Y}^{(h)}_{A,miss}-\widehat{Y}^{(h)}_{A,miss}-\widehat{Y}_A\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\mathbf{M}^{(h)}_i=1} \end{aligned}$$

where

$$\begin{aligned} \widehat{Y}^{(h)}_{A,obs} &= \frac{1}{\widehat{N}^{(h)}_{obs}}\sum_{i=1}^{n} Y_i \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,\mathbf{M}^{(h)}_i=0} \\ \widehat{Y}^{(h)}_{A,miss} &= \frac{1}{\widehat{N}^{(h)}_{miss}}\sum_{i=1}^{n} Y_i \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,\mathbf{M}^{(h)}_i=1} \end{aligned}$$

Thus,

$$\begin{aligned}\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_A\right)^2\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A}=&\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i+\widehat{Y}^{(h)}_{A,obs}\right)^2\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\mathbf{M}_i^{(h)}=0}\\&+\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i+\widehat{Y}^{(h)}_{A,miss}\right)^2\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\mathbf{M}_i^{(h)}=1}\\&+\frac{\widehat{N}^{(h)}_{obs}(A)}{\widehat{N}(A)}\left(\widehat{Y}^{(h)}_{A,obs}-\widehat{Y}_A\right)^2\\&+\frac{\widehat{N}^{(h)}_{miss}(A)}{\widehat{N}(A)}\left(\widehat{Y}^{(h)}_{A,miss}-\widehat{Y}_A\right)^2\end{aligned}$$

Additionally, let us define

$$\begin{aligned}\widehat{N}^{(h)}_{obs}(A_L)&=\sum_{i=1}^{n}\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z,\,\mathbf{M}_i^{(h)}=0}\\\widehat{N}^{(h)}_{miss}(A_L)&=\sum_{i=1}^{n}\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z,\,\mathbf{M}_i^{(h)}=1}\\&\text{(resp. for }\widehat{N}^{(h)}_{obs}(A_R)\text{ and }\widehat{N}^{(h)}_{miss}(A_R))\end{aligned}$$

and

$$\begin{aligned}\widehat{Y}_{A_L,obs}&=\frac{1}{\widehat{N}^{(h)}_{obs}(A_L)}\sum_{i=1}^{n}Y_i\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z,\,\mathbf{M}_i^{(h)}=0}\\\widehat{Y}_{A_L,miss}&=\frac{1}{\widehat{N}^{(h)}_{miss}(A_L)}\sum_{i=1}^{n}Y_i\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z,\,\mathbf{M}_i^{(h)}=1}\\&\text{(resp. for }\widehat{Y}_{A_R,obs}\text{ and }\widehat{Y}_{A_R,miss})\end{aligned}$$

Applying the analogous procedure, we can show that

$$\begin{aligned}&\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_{A_L}\right)^2\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z}\\&=\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i+\widehat{Y}_{A_L,obs}\right)^2\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z,\,\mathbf{M}_i^{(h)}=0}\\&\quad+\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i+\widehat{Y}_{A_L,miss}\right)^2\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,a^{(h)}\leq\widehat{\mathbf{X}}^{(h)}_{i,out}<z,\,\mathbf{M}_i^{(h)}=1}\\&\quad+\frac{\widehat{N}^{(h)}_{obs}(A_L)}{\widehat{N}(A)}\left(\widehat{Y}_{A_L,obs}-\widehat{Y}_{A_L}\right)^2\\&\quad+\frac{\widehat{N}^{(h)}_{miss}(A_L)}{\widehat{N}(A)}\left(\widehat{Y}_{A_L,miss}-\widehat{Y}_{A_L}\right)^2\end{aligned}$$

Using the same ideas for

$$\frac{1}{\widehat{N}(A)}\sum_{i=1}^{n}\left(Y_i-\widehat{Y}_{A_R}\right)^2\mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\,z\leq\widehat{\mathbf{X}}^{(h)}_{i,out}\leq b^{(h)}},$$

the CART criterion can be written as

$$L_n\left(A,d,w\right) = L_{1,n}(A,d) + L_{2,n}\left(A,d,w\right) + L_{3,n}\left(A,d,w\right) + L_{4,n}\left(A,d,w\right)$$

where

$$\begin{aligned}
L_{1,n}(A,d) =& \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A,obs}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, \mathbf{M}_i^{(h)}=0} \\
&- \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_L,obs}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, a^{(h)}\leq \mathbf{X}_i^{(h)} \leq z,\, \mathbf{M}_i^{(h)}=0} \\
&- \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_R,obs}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, z\leq \mathbf{X}_i^{(h)} \leq b^{(h)},\, \mathbf{M}_i^{(h)}=0}
\end{aligned}$$

$$\begin{aligned}
L_{2,n}(A,d,w) =& \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A,miss}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, \mathbf{M}_i^{(h)}=1} \\
&- \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_L,miss}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, a^{(h)}\leq \widehat{\mathbf{X}}_{i,out}^{(h)} \leq z,\, \mathbf{M}_i^{(h)}=1} \\
&- \frac{1}{\widehat{N}(A)} \sum_{i=1}^{n} \left(Y_i - \widehat{Y}_{A_R,miss}\right)^2 \mathbb{1}_{\widehat{\mathbf{X}}_{i,in}\in A,\, z\leq \widehat{\mathbf{X}}_{i,out}^{(h)} \leq b^{(h)},\, \mathbf{M}_i^{(h)}=1}
\end{aligned}$$

$$\begin{aligned}
L_{3,n}(A,d,w) =& \frac{\widehat{N}_{obs}(A)}{\widehat{N}(A)} \left(\widehat{Y}_{A,obs} - \widehat{Y}_A\right)^2 \\
&- \frac{\widehat{N}_{obs}^{(h)}(A_L)}{\widehat{N}(A)} \left(\widehat{Y}_{A_L,obs} - \widehat{Y}_{A_L}\right)^2 \\
&- \frac{\widehat{N}_{obs}^{(h)}(A_R)}{\widehat{N}(A)} \left(\widehat{Y}_{A_R,obs} - \widehat{Y}_{A_R}\right)^2
\end{aligned}$$

$$\begin{aligned}
L_{4,n}(A,d,w) =& \frac{\widehat{N}_{miss}(A)}{\widehat{N}(A)} \left(\widehat{Y}_{A,miss} - \widehat{Y}_A\right)^2 \\
&- \frac{\widehat{N}_{miss}^{(h)}(A_L)}{\widehat{N}(A)} \left(\widehat{Y}_{A_L,miss} - \widehat{Y}_{A_L}\right)^2 \\
&- \frac{\widehat{N}_{miss}^{(h)}(A_R)}{\widehat{N}(A)} \left(\widehat{Y}_{A_R,miss} - \widehat{Y}_{A_R}\right)^2
\end{aligned}$$

B

Tables for the MSE and Bias for Different Approaches varying the Percentage of Missing Values

MAR1

Approach	0	5	10	20	40
No Rows	6.06 ± 0.06	6.84 ± 0.07	7.02 ± 0.07	7.37 ± 0.08	8.35 ± 0.11
No Columns	6.06 ± 0.06	19.79 ± 0.06	19.76 ± 0.06	19.77 ± 0.06	19.76 ± 0.06
Median	6.06 ± 0.06	6.54 ± 0.06	6.57 ± 0.05	6.78 ± 0.06	7.35 ± 0.07
Breiman	6.06 ± 0.06	6.59 ± 0.06	6.64 ± 0.06	6.75 ± 0.06	7.10 ± 0.06
Ishioka	6.06 ± 0.06	6.49 ± 0.06	6.56 ± 0.06	6.72 ± 0.06	7.12 ± 0.07
MissForest	6.06 ± 0.06	6.41 ± 0.06	6.47 ± 0.06	6.49 ± 0.06	6.64 ± 0.06
MIA	6.06 ± 0.06	6.41 ± 0.06	6.47 ± 0.06	6.63 ± 0.06	6.89 ± 0.07
Proposal	6.06 ± 0.06	6.53 ± 0.06	6.56 ± 0.06	6.68 ± 0.06	6.97 ± 0.06

Table B.1: Average mean squared error and its standard error for the different methods, considering the MAR1 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	60	80	90	95
No Rows	10.32 ± 0.14	14.81 ± 0.33	21.07 ± 0.54	26.47 ± 0.87
No Columns	19.75 ± 0.06	19.78 ± 0.06	19.77 ± 0.06	19.77 ± 0.06
Median	8.51 ± 0.09	10.65 ± 0.15	12.24 ± 0.21	14.17 ± 0.28
Breiman	7.90 ± 0.10	9.45 ± 0.13	11.40 ± 0.26	13.70 ± 0.34
Ishioka	7.75 ± 0.08	9.06 ± 0.12	10.23 ± 0.13	11.39 ± 0.20
MissForest	6.97 ± 0.06	7.80 ± 0.09	8.70 ± 0.14	10.32 ± 0.35
MIA	7.42 ± 0.08	8.46 ± 0.11	9.83 ± 0.14	11.13 ± 0.20
Proposal	7.32 ± 0.07	8.11 ± 0.08	8.66 ± 0.08	9.22 ± 0.11

Table B.2: (Cont.). Average mean squared error and its standard error for the different methods, considering the MAR1 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	0	5	10	20	40
No Rows	0.00 ± 0.02	-0.14 ± 0.02	-0.15 ± 0.02	-0.19 ± 0.03	-0.32 ± 0.03
No Columns	0.00 ± 0.02	0.01 ± 0.03	0.01 ± 0.03	0.01 ± 0.03	0.01 ± 0.03
Median	0.00 ± 0.02	-0.06 ± 0.02	-0.08 ± 0.02	-0.09 ± 0.02	-0.20 ± 0.03
Breiman	0.00 ± 0.02	-0.04 ± 0.02	-0.04 ± 0.02	-0.05 ± 0.03	-0.09 ± 0.03
Ishioka	0.00 ± 0.02	-0.04 ± 0.02	-0.05 ± 0.02	-0.07 ± 0.02	-0.16 ± 0.03
MissForest	0.00 ± 0.02	-0.05 ± 0.02	-0.06 ± 0.02	-0.07 ± 0.02	-0.13 ± 0.02
MIA	0.00 ± 0.02	-0.08 ± 0.02	-0.12 ± 0.02	-0.14 ± 0.03	-0.26 ± 0.03
Proposal	0.00 ± 0.02	-0.08 ± 0.02	-0.09 ± 0.02	-0.10 ± 0.02	-0.20 ± 0.03

Table B.3: Average bias and its standard error for the different methods, considering the MAR1 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

Approach	60	80	90	95
No Rows	-0.57 ± 0.04	-1.31 ± 0.08	-1.98 ± 0.11	-2.08 ± 0.16
No Columns	0.00 ± 0.03	-0.02 ± 0.03	-0.01 ± 0.03	-0.02 ± 0.03
Median	-0.35 ± 0.03	-0.67 ± 0.03	-0.73 ± 0.05	-0.82 ± 0.04
Breiman	-0.12 ± 0.03	-0.13 ± 0.03	-0.14 ± 0.03	-0.21 ± 0.05
Ishioka	-0.29 ± 0.03	-0.54 ± 0.03	-0.59 ± 0.05	-0.59 ± 0.05
MissForest	-0.22 ± 0.03	-0.49 ± 0.04	-0.51 ± 0.05	-0.62 ± 0.07
MIA	-0.41 ± 0.03	-0.77 ± 0.03	-0.87 ± 0.05	-0.94 ± 0.05
Proposal	-0.32 ± 0.03	-0.42 ± 0.04	-0.56 ± 0.03	-0.57 ± 0.03

Table B.4: (Cont). Average bias and its standard error for the different methods, considering the MAR1 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

MAR2

Approach	0	5	10	20	40
No Rows	6.06 ± 0.06	6.98 ± 0.07	7.08 ± 0.07	7.47 ± 0.07	8.49 ± 0.10
No Columns	6.06 ± 0.06	19.78 ± 0.06	19.78 ± 0.06	19.75 ± 0.06	19.74 ± 0.06
Median	6.06 ± 0.06	6.52 ± 0.06	6.64 ± 0.06	6.75 ± 0.06	7.44 ± 0.07
Breiman	6.06 ± 0.06	6.54 ± 0.07	6.59 ± 0.06	6.74 ± 0.07	7.23 ± 0.07
Ishioka	6.06 ± 0.06	6.47 ± 0.06	6.57 ± 0.06	6.74 ± 0.07	7.16 ± 0.07
MissForest	6.06 ± 0.06	6.46 ± 0.06	6.48 ± 0.06	6.48 ± 0.06	6.64 ± 0.06
MIA	6.06 ± 0.06	6.41 ± 0.06	6.46 ± 0.06	6.58 ± 0.06	6.98 ± 0.07
Proposal	6.06 ± 0.06	6.59 ± 0.06	6.62 ± 0.06	6.75 ± 0.07	7.05 ± 0.06

Table B.5: Average mean squared error and its standard error for the different methods, considering the MAR2 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	60	80	90	95
No Rows	10.65 ± 0.16	14.71 ± 0.29	18.57 ± 0.40	22.94 ± 0.72
No Columns	19.78 ± 0.06	19.77 ± 0.06	19.77 ± 0.06	19.77 ± 0.06
Median	8.62 ± 0.10	10.45 ± 0.13	12.31 ± 0.22	13.32 ± 0.23
Breiman	8.10 ± 0.09	9.49 ± 0.17	10.79 ± 0.20	12.74 ± 0.27
Ishioka	7.97 ± 0.09	8.82 ± 0.11	9.66 ± 0.14	10.83 ± 0.16
MissForest	7.08 ± 0.07	7.61 ± 0.08	8.34 ± 0.12	9.41 ± 0.25
MIA	7.55 ± 0.09	8.45 ± 0.10	9.18 ± 0.11	10.63 ± 0.18
Proposal	7.52 ± 0.07	7.99 ± 0.08	8.44 ± 0.10	8.93 ± 0.09

Table B.6: (Cont.). Average mean squared error and its standard error for the different methods, considering the MAR2 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	0	5	10	20	40
No Rows	0.00 ± 0.02	-0.19 ± 0.02	-0.20 ± 0.02	-0.25 ± 0.03	-0.46 ± 0.03
No Columns	0.00 ± 0.02	0.02 ± 0.03	0.02 ± 0.03	0.01 ± 0.03	0.01 ± 0.03
Median	0.00 ± 0.02	-0.07 ± 0.02	-0.09 ± 0.02	-0.15 ± 0.02	-0.28 ± 0.02
Breiman	0.00 ± 0.02	-0.03 ± 0.02	-0.05 ± 0.02	-0.08 ± 0.02	-0.08 ± 0.03
Ishioka	0.00 ± 0.02	-0.04 ± 0.02	-0.06 ± 0.02	-0.10 ± 0.02	-0.22 ± 0.02
MissForest	0.00 ± 0.02	-0.05 ± 0.02	-0.06 ± 0.02	-0.10 ± 0.02	-0.16 ± 0.02
MIA	0.00 ± 0.02	-0.08 ± 0.02	-0.12 ± 0.02	-0.17 ± 0.02	-0.36 ± 0.02
Proposal	0.01 ± 0.02	-0.10 ± 0.02	-0.11 ± 0.02	-0.15 ± 0.02	-0.25 ± 0.02

Table B.7: Average bias and its standard error for the different methods, considering the MAR2 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

Approach	60	80	90	95
No Rows	-0.80 ± 0.04	-1.21 ± 0.07	-1.32 ± 0.10	-1.24 ± 0.16
No Columns	0.02 ± 0.03	-0.01 ± 0.03	-0.01 ± 0.03	-0.01 ± 0.03
Median	-0.44 ± 0.03	-0.46 ± 0.05	-0.49 ± 0.04	-0.58 ± 0.03
Breiman	-0.10 ± 0.03	-0.12 ± 0.03	-0.14 ± 0.03	-0.18 ± 0.03
Ishioka	-0.38 ± 0.03	-0.38 ± 0.03	-0.43 ± 0.04	-0.48 ± 0.05
MissForest	-0.30 ± 0.03	-0.36 ± 0.04	-0.39 ± 0.05	-0.41 ± 0.06
MIA	-0.52 ± 0.03	-0.61 ± 0.03	-0.64 ± 0.05	-0.67 ± 0.06
Proposal	-0.28 ± 0.04	-0.38 ± 0.02	-0.43 ± 0.03	-0.47 ± 0.04

Table B.8: (Cont). Average bias and its standard error for the different methods, considering the MAR2 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

MAR3

Approach	0	5	10	20	40
No Rows	6.06 ± 0.06	7.08 ± 0.08	7.07 ± 0.08	7.68 ± 0.09	9.67 ± 0.18
No Columns	6.06 ± 0.06	19.79 ± 0.06	19.74 ± 0.06	19.78 ± 0.06	19.75 ± 0.06
Median	6.06 ± 0.06	6.54 ± 0.06	6.60 ± 0.05	6.96 ± 0.06	7.82 ± 0.08
Breiman	6.06 ± 0.06	6.55 ± 0.06	6.67 ± 0.06	6.84 ± 0.06	7.51 ± 0.08
Ishioka	6.06 ± 0.06	6.63 ± 0.06	6.67 ± 0.06	6.99 ± 0.07	7.73 ± 0.08
MissForest	6.06 ± 0.06	6.74 ± 0.06	6.72 ± 0.06	6.78 ± 0.06	7.02 ± 0.06
MIA	6.06 ± 0.06	6.52 ± 0.06	6.54 ± 0.06	6.81 ± 0.06	7.33 ± 0.06
Proposal	6.06 ± 0.06	6.77 ± 0.06	6.86 ± 0.07	7.05 ± 0.07	7.55 ± 0.07

Table B.9: Average mean squared error and its standard error for the different methods, considering the MAR3 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	60	80	90	95
No Rows	12.74 ± 0.25	17.92 ± 0.44	24.15 ± 0.68	28.28 ± 0.83
No Columns	19.77 ± 0.06	19.80 ± 0.06	19.76 ± 0.06	19.76 ± 0.06
Median	9.19 ± 0.11	10.85 ± 0.15	12.43 ± 0.20	14.05 ± 0.37
Breiman	8.62 ± 0.10	10.00 ± 0.15	11.47 ± 0.23	13.93 ± 0.27
Ishioka	8.64 ± 0.09	9.58 ± 0.10	10.65 ± 0.13	11.95 ± 0.22
MissForest	7.56 ± 0.08	8.23 ± 0.11	9.00 ± 0.15	10.67 ± 0.37
MIA	8.17 ± 0.08	9.11 ± 0.09	10.09 ± 0.12	11.50 ± 0.23
Proposal	8.22 ± 0.08	8.92 ± 0.08	9.28 ± 0.10	9.42 ± 0.10

Table B.10: (Cont.). Average mean squared error and its standard error for the different methods, considering the MAR3 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	0	5	10	20	40
No Rows	0.00 ± 0.02	-0.22 ± 0.03	-0.21 ± 0.03	-0.38 ± 0.03	-0.81 ± 0.04
No Columns	0.00 ± 0.02	0.01 ± 0.03	0.02 ± 0.03	0.01 ± 0.03	0.01 ± 0.03
Median	0.00 ± 0.02	-0.11 ± 0.02	-0.15 ± 0.02	-0.23 ± 0.02	-0.44 ± 0.02
Breiman	0.00 ± 0.02	-0.04 ± 0.02	-0.06 ± 0.02	-0.08 ± 0.02	-0.12 ± 0.02
Ishioka	0.00 ± 0.02	-0.05 ± 0.02	-0.11 ± 0.02	-0.19 ± 0.02	-0.35 ± 0.03
MissForest	0.00 ± 0.02	-0.09 ± 0.02	-0.13 ± 0.02	-0.17 ± 0.02	-0.29 ± 0.02
MIA	0.00 ± 0.02	-0.16 ± 0.02	-0.18 ± 0.02	-0.29 ± 0.02	-0.52 ± 0.03
Proposal	0.00 ± 0.02	-0.17 ± 0.02	-0.19 ± 0.02	-0.27 ± 0.02	-0.43 ± 0.02

Table B.11: Average bias and its standard error for the different methods, considering the MAR3 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

Approach	60	80	90	95
No Rows	-1.33 ± 0.06	-2.00 ± 0.09	-2.62 ± 0.11	-2.53 ± 0.16
No Columns	0.01 ± 0.03	-0.01 ± 0.03	-0.02 ± 0.03	-0.01 ± 0.03
Median	-0.66 ± 0.03	-0.82 ± 0.05	-0.86 ± 0.03	-0.87 ± 0.04
Breiman	-0.11 ± 0.03	-0.11 ± 0.03	-0.15 ± 0.03	-0.23 ± 0.04
Ishioka	-0.56 ± 0.03	-0.60 ± 0.03	-0.60 ± 0.04	-0.70 ± 0.05
MissForest	-0.52 ± 0.03	-0.57 ± 0.04	-0.71 ± 0.06	-0.70 ± 0.06
MIA	-0.78 ± 0.03	-1.00 ± 0.03	-1.01 ± 0.05	-1.13 ± 0.05
Proposal	-0.52 ± 0.04	-0.60 ± 0.03	-0.73 ± 0.03	-0.70 ± 0.04

Table B.12: (Cont). Average bias and its standard error for the different methods, considering the MAR3 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

MAR4

Approach	0	5	10	20	40
No Rows	6.06 ± 0.06	7.31 ± 0.07	7.26 ± 0.08	7.70 ± 0.10	8.89 ± 0.10
No Columns	6.06 ± 0.06	19.76 ± 0.06	19.79 ± 0.06	19.77 ± 0.06	19.79 ± 0.06
Median	6.06 ± 0.06	6.48 ± 0.06	6.58 ± 0.06	6.83 ± 0.06	7.45 ± 0.07
Breiman	6.06 ± 0.06	6.55 ± 0.06	6.58 ± 0.06	6.78 ± 0.06	7.22 ± 0.06
Ishioka	6.06 ± 0.06	6.43 ± 0.06	6.48 ± 0.06	6.61 ± 0.06	7.07 ± 0.08
MissForest	6.06 ± 0.06	6.61 ± 0.06	6.61 ± 0.06	6.65 ± 0.06	6.79 ± 0.06
MIA	6.06 ± 0.06	6.54 ± 0.06	6.58 ± 0.06	6.71 ± 0.06	7.06 ± 0.06
Proposal	6.06 ± 0.06	6.50 ± 0.06	6.53 ± 0.05	6.63 ± 0.06	6.83 ± 0.06

Table B.13: Average mean squared error and its standard error for the different methods, considering the MAR4 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	60	80	90	95
No Rows	10.80 ± 0.15	14.35 ± 0.24	17.79 ± 0.30	22.65 ± 0.83
No Columns	19.78 ± 0.06	19.77 ± 0.06	19.77 ± 0.06	19.77 ± 0.06
Median	8.59 ± 0.10	10.25 ± 0.14	12.49 ± 0.23	13.31 ± 0.27
Breiman	8.01 ± 0.09	8.98 ± 0.11	10.82 ± 0.25	12.72 ± 0.37
Ishioka	7.57 ± 0.07	8.17 ± 0.08	9.11 ± 0.14	10.52 ± 0.15
MissForest	7.09 ± 0.06	7.54 ± 0.06	8.25 ± 0.12	9.48 ± 0.37
MIA	7.58 ± 0.08	8.12 ± 0.08	8.92 ± 0.13	10.19 ± 0.23
Proposal	7.20 ± 0.06	7.62 ± 0.07	8.07 ± 0.07	8.71 ± 0.10

Table B.14: (Cont.). Average mean squared error and its standard error for the different methods, considering the MAR4 case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	0	5	10	20	40
No Rows	0.00 ± 0.02	0.28 ± 0.03	0.27 ± 0.03	0.27 ± 0.03	0.25 ± 0.03
No Columns	0.00 ± 0.02	0.01 ± 0.04	0.01 ± 0.03	0.01 ± 0.03	0.01 ± 0.03
Median	0.00 ± 0.02	0.04 ± 0.02	0.09 ± 0.02	0.15 ± 0.02	0.14 ± 0.02
Breiman	0.00 ± 0.02	0.09 ± 0.02	0.08 ± 0.02	0.08 ± 0.02	0.12 ± 0.02
Ishioka	0.00 ± 0.02	-0.01 ± 0.02	0.05 ± 0.02	0.08 ± 0.02	0.09 ± 0.02
MissForest	0.00 ± 0.02	0.05 ± 0.02	0.07 ± 0.02	0.08 ± 0.02	0.09 ± 0.02
MIA	0.00 ± 0.02	0.12 ± 0.02	0.17 ± 0.02	0.24 ± 0.02	0.26 ± 0.02
Proposal	0.00 ± 0.02	0.04 ± 0.02	0.04 ± 0.02	0.13 ± 0.02	0.13 ± 0.02

Table B.15: Average bias and its standard error for the different methods, considering the MAR4 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

Approach	60	80	90	95
No Rows	0.29 ± 0.05	0.24 ± 0.08	0.23 ± 0.10	0.56 ± 0.15
No Columns	0.00 ± 0.03	-0.02 ± 0.03	-0.02 ± 0.03	-0.02 ± 0.03
Median	0.14 ± 0.03	0.15 ± 0.03	0.15 ± 0.04	0.26 ± 0.05
Breiman	0.11 ± 0.03	0.11 ± 0.03	0.10 ± 0.03	0.10 ± 0.04
Ishioka	0.10 ± 0.03	0.11 ± 0.03	0.11 ± 0.04	0.12 ± 0.06
MissForest	0.10 ± 0.03	0.10 ± 0.04	0.14 ± 0.05	0.23 ± 0.07
MIA	0.24 ± 0.03	0.25 ± 0.04	0.26 ± 0.05	0.30 ± 0.06
Proposal	0.13 ± 0.03	0.13 ± 0.03	0.13 ± 0.04	0.16 ± 0.03

Table B.16: (Cont). Average bias and its standard error for the different methods, considering the MAR4 case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

LOG

Approach	0	5	10	20	40
No Rows	6.06 ± 0.06	6.85 ± 0.08	6.95 ± 0.08	7.38 ± 0.09	8.20 ± 0.10
No Columns	6.06 ± 0.06	19.76 ± 0.06	19.79 ± 0.06	19.76 ± 0.06	19.76 ± 0.06
Median	6.06 ± 0.06	6.45 ± 0.06	6.53 ± 0.07	6.74 ± 0.07	7.24 ± 0.08
Breiman	6.06 ± 0.06	6.49 ± 0.06	6.54 ± 0.06	6.69 ± 0.07	7.09 ± 0.08
Ishioka	6.06 ± 0.06	6.35 ± 0.06	6.42 ± 0.06	6.60 ± 0.06	6.92 ± 0.07
MissForest	6.06 ± 0.06	6.37 ± 0.06	6.38 ± 0.06	6.42 ± 0.06	6.55 ± 0.06
MIA	6.06 ± 0.06	6.36 ± 0.06	6.42 ± 0.06	6.53 ± 0.06	6.74 ± 0.07
Proposal	6.06 ± 0.06	6.41 ± 0.06	6.47 ± 0.06	6.54 ± 0.06	6.79 ± 0.07

Table B.17: Average mean squared error and its standard error for the different methods, considering the LOG case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	60	80	90	95
No Rows	9.67 ± 0.13	12.81 ± 0.20	17.68 ± 0.42	22.66 ± 0.65
No Columns	19.78 ± 0.06	19.77 ± 0.06	19.78 ± 0.06	19.78 ± 0.06
Median	8.15 ± 0.08	10.12 ± 0.15	12.21 ± 0.22	13.42 ± 0.25
Breiman	7.58 ± 0.08	8.96 ± 0.16	10.77 ± 0.24	12.48 ± 0.31
Ishioka	7.34 ± 0.08	8.13 ± 0.10	9.39 ± 0.15	10.63 ± 0.19
MissForest	6.76 ± 0.06	7.30 ± 0.07	8.25 ± 0.11	9.51 ± 0.25
MIA	7.11 ± 0.08	7.84 ± 0.10	8.95 ± 0.15	10.22 ± 0.18
Proposal	6.92 ± 0.07	7.36 ± 0.07	8.01 ± 0.08	8.88 ± 0.09

Table B.18: (Cont.). Average mean squared error and its standard error for the different methods, considering the LOG case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	0	5	10	20	40
No Rows	0.00 ± 0.02	-0.04 ± 0.02	-0.04 ± 0.03	-0.08 ± 0.03	-0.11 ± 0.03
No Columns	0.00 ± 0.02	0.01 ± 0.03	0.02 ± 0.03	0.01 ± 0.03	0.02 ± 0.03
Median	0.00 ± 0.02	0.01 ± 0.02	0.00 ± 0.02	-0.01 ± 0.02	-0.05 ± 0.02
Breiman	0.00 ± 0.02	0.01 ± 0.02	0.01 ± 0.02	-0.01 ± 0.02	-0.03 ± 0.03
Ishioka	0.00 ± 0.02	0.01 ± 0.02	0.00 ± 0.02	-0.02 ± 0.02	-0.03 ± 0.03
MissForest	0.00 ± 0.02	0.00 ± 0.02	0.00 ± 0.02	0.00 ± 0.02	-0.04 ± 0.02
MIA	0.00 ± 0.02	0.03 ± 0.02	0.02 ± 0.02	-0.01 ± 0.03	-0.07 ± 0.03
Proposal	0.01 ± 0.02	-0.01 ± 0.02	-0.01 ± 0.02	-0.03 ± 0.02	-0.08 ± 0.02

Table B.19: Average bias and its standard error for the different methods, considering the LOG case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

Approach	60	80	90	95
No Rows	-0.14 ± 0.04	-0.30 ± 0.07	-0.72 ± 0.11	-0.97 ± 0.17
No Columns	0.00 ± 0.03	-0.02 ± 0.03	-0.02 ± 0.03	-0.02 ± 0.03
Median	-0.09 ± 0.03	-0.19 ± 0.03	-0.26 ± 0.05	-0.28 ± 0.05
Breiman	-0.03 ± 0.03	-0.04 ± 0.03	-0.07 ± 0.03	-0.13 ± 0.04
Ishioka	-0.08 ± 0.03	-0.17 ± 0.03	-0.23 ± 0.05	-0.28 ± 0.05
MissForest	-0.06 ± 0.03	-0.11 ± 0.03	-0.26 ± 0.06	-0.29 ± 0.07
MIA	-0.10 ± 0.03	-0.24 ± 0.04	-0.32 ± 0.05	-0.35 ± 0.06
Proposal	-0.09 ± 0.03	-0.16 ± 0.03	-0.25 ± 0.04	-0.22 ± 0.04

Table B.20: (Cont). Average bias and its standard error for the different methods, considering the LOG case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

DEPY

Approach	0	5	10	20	40
No Rows	6.06 ± 0.06	7.44 ± 0.09	7.79 ± 0.08	8.49 ± 0.12	10.81 ± 0.18
No Columns	6.06 ± 0.06	19.76 ± 0.06	19.78 ± 0.06	19.78 ± 0.06	19.75 ± 0.05
Median	6.06 ± 0.06	6.77 ± 0.06	6.82 ± 0.06	7.15 ± 0.07	7.87 ± 0.08
Breiman	6.06 ± 0.06	6.75 ± 0.06	6.92 ± 0.07	7.05 ± 0.07	7.76 ± 0.08
Ishioka	6.06 ± 0.06	6.70 ± 0.07	6.88 ± 0.07	7.26 ± 0.07	8.14 ± 0.09
MissForest	6.06 ± 0.06	6.58 ± 0.06	6.57 ± 0.06	6.68 ± 0.06	6.87 ± 0.07
MIA	6.06 ± 0.06	6.76 ± 0.07	6.93 ± 0.07	7.23 ± 0.08	7.80 ± 0.08
Proposal	6.06 ± 0.06	6.59 ± 0.06	6.67 ± 0.06	6.86 ± 0.06	7.37 ± 0.07

Table B.21: Average mean squared error and its standard error for the different methods, considering the DEPY case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	60	80	90	95
No Rows	16.37 ± 0.29	23.87 ± 0.26	28.53 ± 0.41	31.78 ± 0.56
No Columns	19.75 ± 0.06	19.77 ± 0.06	19.77 ± 0.06	19.77 ± 0.06
Median	9.86 ± 0.13	13.67 ± 0.20	15.43 ± 0.23	16.01 ± 0.26
Breiman	9.41 ± 0.12	13.10 ± 0.20	15.46 ± 0.27	16.07 ± 0.25
Ishioka	10.04 ± 0.13	12.18 ± 0.14	13.07 ± 0.19	13.57 ± 0.22
MissForest	7.75 ± 0.08	9.32 ± 0.16	10.66 ± 0.31	12.62 ± 0.54
MIA	9.47 ± 0.12	11.72 ± 0.15	12.86 ± 0.16	13.97 ± 0.27
Proposal	8.57 ± 0.11	8.88 ± 0.10	9.05 ± 0.09	9.07 ± 0.09

Table B.22: (Cont.). Average mean squared error and its standard error for the different methods, considering the DEPY case. The three methods with the lower MSE are filled in blue, while the three methods with the highest MSE are filled in orange.

Approach	0	5	10	20	40
No Rows	0.00 ± 0.02	0.46 ± 0.02	0.55 ± 0.03	0.70 ± 0.03	1.09 ± 0.03
No Columns	0.00 ± 0.02	0.01 ± 0.04	0.01 ± 0.04	0.02 ± 0.03	0.02 ± 0.03
Median	0.00 ± 0.02	0.11 ± 0.02	0.12 ± 0.02	0.19 ± 0.02	0.37 ± 0.02
Breiman	0.00 ± 0.02	0.04 ± 0.02	0.04 ± 0.02	0.07 ± 0.02	0.08 ± 0.03
Ishioka	0.00 ± 0.02	0.06 ± 0.02	0.06 ± 0.02	0.12 ± 0.02	0.23 ± 0.03
MissForest	0.00 ± 0.02	0.06 ± 0.02	0.05 ± 0.02	0.08 ± 0.02	0.11 ± 0.02
MIA	0.00 ± 0.02	0.37 ± 0.02	0.40 ± 0.02	0.54 ± 0.02	0.78 ± 0.03
Proposal	0.00 ± 0.02	0.20 ± 0.02	0.22 ± 0.02	0.29 ± 0.02	0.34 ± 0.04

Table B.23: Average bias and its standard error for the different methods, considering the DEPY case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.

Approach	60	80	90	95
No Rows	1.80 ± 0.04	2.54 ± 0.04	2.82 ± 0.07	2.95 ± 0.09
No Columns	0.02 ± 0.03	-0.03 ± 0.03	-0.01 ± 0.03	-0.02 ± 0.03
Median	0.67 ± 0.03	1.03 ± 0.03	1.05 ± 0.03	1.04 ± 0.05
Breiman	0.12 ± 0.03	0.14 ± 0.03	0.17 ± 0.03	0.24 ± 0.04
Ishioka	0.45 ± 0.03	0.69 ± 0.03	0.75 ± 0.03	0.74 ± 0.04
MissForest	0.20 ± 0.03	0.39 ± 0.03	0.39 ± 0.03	0.48 ± 0.06
MIA	1.13 ± 0.03	1.44 ± 0.03	1.38 ± 0.03	1.29 ± 0.04
Proposal	0.44 ± 0.03	0.55 ± 0.04	0.69 ± 0.03	0.77 ± 0.03

Table B.24: (Cont). Average bias and its standard error for the different methods, considering the DEPY case. The three more unbiased methods are filled in blue, while the three more biased methods are filled in orange.