Performance and accessibility of statistical learning algorithms for applied data analysis

by

Cédric Beaulac

A thesis submitted in conformity with the requirements
for the degree of Doctor of Statistics
Graduate Department of Statistical Sciences
University of Toronto

Abstract

Performance and accessibility of statistical learning algorithms for applied data analysis

Cédric Beaulac
Doctor of Statistics
Graduate Department of Statistical Sciences
University of Toronto
2021

In this thesis we explore a wide range of statistical learning algorithms and evaluate their abilities to answer clear and precise research questions given a real data set. We do so in multiple research fields; we tackle a higher education problem, contribute to oncology research and analyse an image data set. Our evaluation of algorithms is made with respect to their performance on these real data sets and their overall accessibility. This creates opportunities for interesting findings and results and it serves as motivation for theoretical and algorithmic contributions. We revisit the theoretical foundation of algorithms in order to adapt them to tackle well-established statistical problems. We also visit every step of data analysis; from data collection passing by algorithm design to a thorough analysis that answers research questions. This gives us a complete perspective of the entire data analysis pipeline which allows us to form constructive criticisms about algorithms. In this thesis we discuss all of our recent contributions. We analyse a higher education data set and demonstrate the ability of algorithms to accurately predict students in needs of support. Inspired by the structure of this data set, we construct a new decision algorithm and provide a R-package to anyone interested in using this algorithm. We study the gap between the theory and the implementation of variational autoencoders, illustrate these differences on real data sets and provide explanations and recommendations on how to fix these issues. We design a new survival analysis model using the variational autoencoder framework and evaluate multiple survival analysis machine learning models on the largest randomized trial in pediatric Hodgkin Lymphoma ever conducted. We collect and share a new data set containing high-resolution images of hand-written digits attached to a collection of covariates. Finally, we demonstrate the potential of controllable generative models; a new perspective on the use of labelled data sets and generative models to provide user control over the generated variables.

Acknowledgements

To begin I would like to thank my supervisor Jeffrey S. Rosenthal for the tremendous support he offered me throughout the last few years. You were reassuring, calming, optimistic and provided me with insightful comments on every project I worked on. I enjoyed our conversations a lot and I am going to miss you. Merci beaucoup!

I also want to thank David Duvenaud for guiding me into the world of Machine Learning. I would not have been able to do any of my recent work without your help. Our conversations about the differences between Statistics and Machine Learning really sparked something and I am still thinking about those conversations regularly. You were always pushing me to do better and I will always be grateful for that. Thank you very much.

I want to thank every member of my thesis evaluation committee including Patrick Brown, Stanislav Volgushev and Dehan Kong for the constructive comments and questions that inspired the last waves of corrections included in the thesis.

Finalement, je voudrais remercier mes parents, ma fiancée et tous mes amis qui m'ont supporté tout au long de ce processus très long et très éprouvant. Je n'y serais jamais arrivé sans votre aide. Je vous aime. Merci.

Contents

Chapter 1

Introduction

1.1 Preface

When I was a kid, I was fascinated by Artificial Intelligence (AI). At that point in time I was thinking about independent-minded robots, self-driving cars and realistic non-playable characters in video games. Back then it seemed unrealistic but we are currently allowed to dream about these things; these things are now within arm's reach. The recent evolution of automated decision-making is grounded in Machine Learning (ML), an algorithmic approach to data analysis.

As someone who studied statistics for years, I was intrigued to study what made those fields different. If I was to define statistics as a probabilistic approach to data analysis, we could see a difference on sight. However, this oversimplifies the difference too much; machine learning is built on probability results and statistics relies on algorithms in many cases.

Both fields are trying to accomplish similar goals and, from experience, it feels like the main difference are the different research communities. Being a member of both research communities felt important to me and producing research that would be recognized as contributions for both communities was something I tried to achieves throughout my Ph.D.

1.2 Content of the thesis

In this thesis we approach ML with an applied data analysis perspective. We discuss popular ML models, the noticeable differences with commonly used statistical techniques and discuss performances for various tasks on simulated and real data sets.

Statistical modeling: The two cultures [21] was very influential for me and this thesis is an attempt at better understanding how to utilize both statistics models and machine learning models to solve real data problems. Uniting these two research communities is an important step and one we wish to accomplish. To do so, it is important that we understand the strengths and weaknesses of both of these different approaches.

The common theme in all of our research projects is the evaluation of modern machine learning model contributions to the current state of data analysis. This evaluation is done with respect to

the performances and the accessibility of the visited algorithms. When evaluating the accessibility of algorithms, we discuss the availability of implementations, packages and libraries, the computing power needed to use those and the data assumption under which they were built. When evaluating performances we not only consider measures of accuracy and error but we also discuss generalization abilities and the stability of algorithms.

To submerge ourselves further in machine learning, we explored in the entire machine learning pipeline. We delved into data gathering and processing, algorithm design, data analysis, surveys and more.

1.2.1 Contributions

The following is a list of all novel contributions sorted by order of appearance in the thesis.

- An analysis of a data set provided by the University of Toronto using Random Forests; the analysis focus on the prediction of program and program completion and the important predictors.
- A novel Decision Tree algorithm, BESTree (R-package) that allows researchers to guide the partitioning process. We illustrated how to utilize our novel algorithm to analyze data with missing values.
- An analysis of the current Variational AutoEncoder (VAE) implementations; we discuss the differences between the proposed VAE model and successful implementations and support our arguments with experiments on the MNIST data set.
- A novel VAE model called Survival Analysis VAE (SAVAE); this model was designed to adapt VAEs for survival analysis.
- An extensive survey of survival analysis machine learning techniques; the survey was done on the largest randomized trial in pediatric Hodgkin Lymphoma ever conducted.
- A new computer vision data set; inspired by MNIST, this data set contains 13580 hand written digit images in much higher resolution (500×500 pixels). The data also contains writer identification and various writer characteristics.
- An analysis of this brand new data set; we explore a wide range of supervised, unsupervised and semi-supervised tasks.
- A demonstration of controllable generative algorithms that allows users to steer the generative procedure using the data set we propose; we train a generative model on our new proposed data set and are able to generate images that mimics a writing style.

1.2.2 Organization

The next chapter is dedicated to briefly introduce a wide range of ML models. In Chapter 3 we will introduce our first research article; a data analysis application of random forest. In Chapter 4 we will introduce a novel decision tree algorithm created and tested with traditional ML methods which lead to our second publication. In Chapter 5 we discuss some of the problems with the simple VAE model

initially proposed. Then, in Chapter 6, we introduce a survey of the performances of ML techniques in a traditional statistical field: survival analysis. This led to our third submission. Next, Chapter 7 discusses our most recent project which introduces our newly created data set and our experiments with computer vision algorithms. Finally, Chapter 8 concludes this thesis with a short discussion.

Chapter 2

Machine learning background

In this chapter we introduce multiple machine learning (ML) algorithms and some notation used throughout this thesis. This chapter is meant as a brief overview rather than than a thorough definition. Readers comfortable with ML terminologies may skip this chapter.

Most of the theory explained in this chapter and the notation come from both Shalev-Shwartz and Ben-David *Understanding Machine Learning: From Theory to Algorithms* [133] and Hastie et al. *The Elements of Statistical Learning* [61]. This is the theory upon which we have built most of this thesis.

2.1 Supervised learning

In supervised learning we are interested in the relationship between a set of variables called predictors, inputs or explanatory variables and another set of variables called responses, outputs or explained variables. The problem usually consists of learning ways to predict the responses when given predictors.

We identify the set of predictor variable as $\mathbf{x}$, say of size m, and the set of response variable as $\mathbf{y}$, say of size t and a full observation as $\mathbf{z} = \{\mathbf{x}, \mathbf{y}\}$ of size $m + t$. Each of these variables belong to their own respective spaces: $\mathbf{x}$ belong to the predictor space $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_m$, the response variables in $\mathcal{Y} = \mathcal{Y}_1 \times ... \times \mathcal{Y}_t$ and $\mathbf{z}$ belong to $\mathcal{X} \times \mathcal{Y} = \mathcal{Z}$.

Additionally, it is common to receive a data set $S = \{\mathbf{x}_i, \mathbf{y}_i : i \in (1, ..., n)\}$ containing n different observations of the predictors and the responses. We are tasked to use this data set S, the training set, to produce a prediction function h that takes new predictors $\mathbf{x}$ as input and returns a prediction $h_S(\mathbf{x}) = \hat{\mathbf{y}}$. Notice that we have indexed the prediction function h by the data set S, since different training sets lead to different prediction functions. To keep the notation clean we will exclude the index S when it is not necessary.

A classic assumption in supervised learning is that the training set S is a sample of observations that are independently distributed according to the same unknown and true distribution $\mathcal{D}$, i.e. $\mathbf{Z} = \mathbf{X} \times \mathbf{Y} \sim \mathcal{D}$. Thus, the problem becomes one of approximating the true distribution $\mathcal{D}$ as best as possible, more precisely in supervised learning we want to approximate $\mathcal{D}(\mathbf{Y}|\mathbf{X})$.

2.1.1 Loss function

If the response $\mathbf{y}$ is a categorical variable we talk about classification problems and if the response is continuous we talk about regression. To introduce some theoretical concepts, let us begin by discussing the classification problem given a data set with a single response y. The true loss of a classifier h is defined as

$$L_{\mathcal{D}}(h) = \mathbf{P}_{\mathcal{D}}[h(\mathbf{x}) \neq y], \tag{2.1}$$

which is the probability under the true data generating distribution $\mathcal{D}$ that the classifier h misclassifies an observation $\mathbf{x}$. Thus we can claim that we have achieved our goal of estimating the true data distribution $\mathcal{D}$ if the loss is zero. Since the data generating distribution $\mathcal{D}$ is unknown, we cannot compute the true loss and the empirical loss computed with the data set S is typically used as an estimator

$$L_S(h) = \frac{|\{i \in [n] : h(\mathbf{x}_i) \neq y_i\}|}{n}, \tag{2.2}$$

which is the proportion of misclassified observations in the training set S. To establish a prediction function we first define a large set of possible classifiers $\mathcal{H}$, the hypothesis class, and then we try to select $h \in \mathcal{H}$ that minimizes the empirical loss function.

To extend the above definition to a wider range of learning tasks we need to generalize this definition. For instance, the true loss of Equation 2.1 would not be informative if y was continuous. To generalize this concept, we define *loss functions* $l : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}^+$ and the loss of the classifier as

$$\mathcal{L}_D(h) = \mathbb{E}_{\mathcal{D}}[l(h, \mathbf{z})]. \tag{2.3}$$

Consequently, we define the empirical loss as

$$L_S(h) = \frac{\sum_{i=1}^{n} l(h, \mathbf{z}_i)}{n}. \tag{2.4}$$

To respect the intuition of the first definition of Equation 2.1, we usually create loss functions such that a big loss is associated with a poor fit and a small loss with a good fit. For instance, if y is a categorical a simple loss function is the 0-1 loss

$$l_S(h) = \mathbf{1}(h(\mathbf{x}_i) = y_i). \tag{2.5}$$

As a matter of fact, under this loss function both the true loss and empirical loss are back to their original forms expressed in Equations 2.1 and 2.2. With this generalized definition of loss we can create loss function that works well for continuous response such as the squared error (SE) loss

$$l_S(h) = (h(\mathbf{x}_i) - y_i)^2, \tag{2.6}$$

and in this case, the empirical loss is the mean squared error (MSE), a very well-known loss

$$L_S(h) = \frac{\sum_{i=1}^{n} (h(\mathbf{x}_i) - y_i)^2}{n}. \tag{2.7}$$

2.1.2 Decision tree

A classifier h is built to emit a class prediction for any new data point $\mathbf{x}$ that belongs in the predictor space $\mathcal{X} = \mathcal{X}_1 \times ... \times \mathcal{X}_m$. Assuming the response variable can take k different values, the classifier divides the predictor space $\mathcal{X}$ into k disjoint regions $K_1, ...K_k$, one per class, such that $\cup_{q=1}^{k} K_q = \mathcal{X}$, i.e. $h(\mathbf{x}) = \sum_{q=1}^{k} q\mathbf{1}\{\mathbf{x} \in K_q\}$.

A classification tree [22], or Decision Tree (DT), is an algorithm that directly forms regions in the predictor space by recursively dividing it. In this section we will quickly introduce one of the simplest form of decision trees, a binary classification tree has defined by Breiman et al. [22]. A binary classification tree is obtained through a sequence of recursive binary partitioning of the predictor space. Beginning with the entire predictor space, the algorithm selects the variable to split upon and the data partition that minimize some impurity measure. Then the resulting two regions are each split into two more regions until some stopping rule is applied. The classifier will label each region with one of the k possible classes. It is possible to label multiple regions with the same class. For instance, if R_1 and R_2 are both assigned the label l then $K_l = R_1 \cup R_2$. Since decision trees can be graphically represented with nodes and edges we use the word *node* interchangeably with regions (R) of the predictor space.

It is said that the four elements needed for the binary tree growing procedure are:

1. A set of binary questions.
2. A goodness of split criterion (usually resulting in minimizing some impurity measure)
3. A stop-splitting rule
4. A terminal node (leaf) labelling rule

There exist multiple ways to establish these four *ingredients* but in order to keep this introduction succinct we only introduce a few simple ways to define them.

A simple labelling process (4) goes as follows; let $p_{rq} = \frac{1}{n_r} \sum_{\mathbf{x}_i \in R_r} \mathbf{1}\{y_i = q\}$, the proportion of the class q in the region r where n_r is the number of observations contained in region r. Then, the label of the region r is the majority class in that region, i.e. if $\mathbf{x} \in R_r$, $h_S(x) = \text{argmax}_q(p_{rq})$. The impurity measure function for region r is defined as Q_r (2) and can take many forms such as the *Gini index* or the *deviance*

$$\begin{aligned} \text{Gini index}: \ Q_r &= \sum_{q=1}^{k} p_{rq}(1 - p_{rq}) \\ \text{Deviance}: \ Q_r &= -\sum_{q=1}^{k} p_{rq} \log(p_{rq}). \end{aligned} \tag{2.8}$$

When splitting a region R_p into two new regions R_r and R_t the search algorithm computes the total impurity of the new regions ; $n_r Q_r + n_t Q_t$ and picks the split variable x_j and the split s that minimizes that total impurity. We establish the set of binary questions (1) differently depending on the nature of the predictors. If the predictor x_j is continuous, the possible splits are of the form $x_j \leq s$ and $x_j > s$

which usually results in $n_p - 1$ possible splits. For a categorical predictor having c possible values, we usually consider all of the $2^{c-1} - 1$ possible partitions.

The partitioning continues until a stopping rule is applied (3). In some cases, the algorithm stops whenever every terminal node of the tree contains less than β observations, in other cases it stops when all observations within a region belong to the same class. To prevent overfitting, a deep tree is built and then the tree is pruned. Tree-pruning is a cost-complexity procedure that relies on considering that each terminal node is associated with a cost α. The procedure begins by collapsing leaves that produce the smallest increase in total impurity and this technique will collapse leaves as long as the increase in impurity is less than the cost α of the additional leaf. Rigorously, the tree pruning procedure consists of minimizing the cost complexity criterion defined as

$$C_\alpha = \sum_{r=1}^{|R|} n_r Q_r + \alpha |R|, \tag{2.9}$$

where $|R|$ is the number of regions, i.e. the number of leaves in the tree. As α increases, a unique nested set of trees is generated as solutions of Equation 2.9. The parameter α can be determined by cross-validation or with the use of a validation set.

To summarize, here is a pseudo-code of a simple DT as described above used as a visual support.

Algorithm 1 : DT(S,β)

INPUT: training set S and an hyper-parameter β
Start with the entire data set S as the node r
1 **if** All labels are the same :
  Assign the label to the node and exit.
if $n_r < \beta$:
  Assign the majority label to the node and exit.
else :
  for j in all predictors:
    for s in all possible splits :
      Compute the total impurity measure for the two resulting regions.
  Select the variable j and the split s with minimum total impurity measure.
  Split the node into two child nodes.
  Repeat step 1 on the two children nodes.
OUTPUT: A fitted binary decision tree

Decision trees can also be extended for continuous responses. In this case we talk about regression trees. The labelling rule could be the output mean within the terminal node and the impurity measure could be the MSE for instance.

To conclude this section, let us discuss why and how DTs are part of our research. When reading statistical learning books [61, 15], the decision tree was the first model that felt drastically different. It is drastically different from all statistical techniques we have been in contact with before. The classification problem is framed as an optimization problem and the solution is strictly algorithmic. It assumes no

model nor particular structure. Nonetheless, it produces a highly interpretable prediction function which is important in the eyes of many applied data scientists and is usually a characteristic attributed to linear statistical models. For these reasons, we spent some time on decision trees; we have explored their strengths and weaknesses in our first few projects. We used decision trees for a wide range of applied problems. We also came up with a new decision algorithm which will be discussed in Chapter 4. However, decision trees greatly suffer from instability which we discuss next.

2.1.3 Random forests

Random forests and bagged trees proved to be a great improvement, having higher accuracy and lower variance, over decision trees due to the instability of decision trees. Thus it feels natural to briefly introduce the concept of algorithm stability.

Before coming up with the concept of bagging (**B**oostrap **agg**regat**ing**) and random forests, Breiman studied exhaustively instability in a 1996 publication [19]. A heuristic definition of instability is that a small change in the training set S can make a big change in the resulting prediction function h_S. Thus, the more unstable the procedure is, the noisier the empirical loss is and consequently it is harder to select the best prediction function h, the one with the minimal true loss, within the hypothesis class $\mathcal{H}$.

A more rigorous definition of algorithm stability is established by Shalev-Shwartz and Ben-David [133]. Given a training set S and an additional observation $z^* = (\mathbf{x}^*, \mathbf{y}^*)$ we can define a sequence of training sets $S^{(i)}$ where the ith observation is replaced by z^*. This allows us to precisely define what a *small change in the training set* is. In this case it means using $S^{(i)}$ instead of S when training the prediction function h. To quantify the effect of the change in the training on the prediction function we use a loss function l and compute $l(h_S^{(i)}, z_i) - l(h_S, z_i) \geq 0$. The smaller this value is the more stable the procedure is. Shalev-Shwartz and Ben-David [133] later prove how stable algorithms do not overfit. In other words, stable models generalize well to new observations.

Decision trees were empirically shown to be unstable by Breiman [19]. Bagged tree [18] was first proposed as a solution to this problem and later Random forest [20] came as an updated version of bagged tree. We consider a random forest to be any ensemble of tree prediction functions but the name was first coined to represent Breiman's implementation introduced in 2001 [20]. Let us now introduce these models.

Bagging relies on the fact that aggregating the predictions of a large ensemble of prediction functions is more stable and accurate than those individual prediction functions themselves. Full proof is included in [18]. A problem is that we rarely have multiple training set S to allow us to build multiple classifiers h_S. The solution proposed by Breiman is to take T bootstrap samples S^* of the training S, say $S_1^*, S_2^*, ..., S_T^*$ and fit a decision tree using each of these samples $h_{S_1^*}, ..., h_{S_T^*}$. For prediction we aggregate the multiple tree by taking the average of the predicted value if y is continuous. If y is categorical, then each tree predict a class for y and each of these predicted classes are considered as a vote towards the final prediction where the majority wins. This technique is known as bagged trees and is the simplest form of Random Forests.

Here is a pseudo-code on how to form bagged trees:

Algorithm 2 : Bagged Trees(S,β,T)
INPUT: training set S, hyper-parameter β and the number of trees in the ensemble T. 1 Take T bootstrap samples of S resulting in $S_1^*, ..., S_T^*$. 2 **for** all t in (1,...,T): Fit DT : $h_t = \text{DT}(S_t,\beta)$ OUTPUT: An ensemble of DT prediction functions.

To further improve the stability of the ensemble of trees, Breiman proposed a technique to ensure that the produced trees are as uncorrelated as possible [20], this leads to the most popular implementations of Random Forest (RF). It is shown [20] that having individual classifiers that are as uncorrelated as possible further increase the stability of the ensemble. Breiman defines the correlation between classifiers has a function of the random vectors leading to the classifiers. For bagged trees, those random vector are the random bootstrap samples. However when building a RF, these random vectors are more complicated. Once again we take T bootstrap samples S^* of the training S however ,during the partitioning process instead of looking for the best split among all predictors, a subset of predictors is randomly sampled. This random sample is also part of the random vector leading to the classifier, and thus the extra layer of randomness in RF directly leads to a forest of trees less correlated compared to the previously described bagged trees.

Beside being used as a classification or regression algorithms, RFs became quite popular for exploratory data analysis where they are used to compute the variable importance. A variable importance analysis aims at understanding the effect of individual predictors on the classifier outcome. More precisely, a variable importance analysis orders the list of predictors by how much they individually affect the random forest. A predictor with a great effect is considered an important predictor. A RF provides multiple variable importance computations.

The permutation decrease importance was introduced by Breiman along with RFs [20]. Intuitively, if a predictor has a significant effect on the response, perturbing this predictor during prediction should result in a lower accuracy. One way to disrupt the predictor values is by permutation. The permutation decrease importance is a procedure that computes the prediction accuracy on the test set first. Then, it permutes the test set observations of one predictor, say x_j, run this permuted data through the forest and compute the accuracy again. The process is repeated for all predictors and the prediction accuracy decreases are compared. The larger the decrease in accuracy the more important the variable is considered. Through the years new variable importance techniques were established and we discuss them in later chapters.

Variable importances is a novelty for me as a statistician as it was motivated by black-box models. Neither the p-value nor the size of the coefficient itself is a good candidate to represent to *importance* of a predictor. For instance, a predictor could be significant according to its p-value but be attached to a coefficient small in comparison to the scale of the responses variable to a point that the predictor has close to no effect on the response. The opposite case where the coefficient is large but considered non-statistically significant is also problematic when we try to asses the impact of the predictor on the

response. The variable importance analysis built along RFs puts the quality of the predictions at the center of the evaluation of a predictor's importance and has been used successfully in the literature [139].

However, this does not mean that variable importance computations are exclusive to RFs. As a matter of fact, the permutation decrease importance is model agnostic and there exist R packages and Python libraries that allow users to compute the permutation decrease importance for a wide range of models. Similarly, some importance techniques have been designed for linear models, for instance the dominance analysis [23] compares the contribution to the coefficient of determination, R^2, of multiple predictors. Nevertheless, the decrease importance is a significant contribution has it defines a way to explore the mechanism of black-box models such as RFs.

RFs were at the centre of our very first research project and they were used notably to perform a variable importance analysis on real data. Regarding our study of ML, bagging is a great example of statistically motivated ideas implemented on a unstable algorithmic approach, DTs. The trade-off between the high interpretability but instability of DTs and their more stable but non-interpretable RF counterpart was influential for our understanding on how some ML models are established; instability was empirically motivated but the solution was theoretically established.

2.1.4 Neural networks

Neural Networks (NNs) are central to the increase in popularity of ML in the recent years. NNs are flexible parametric functions composed of parametric linear and fixed non-linear transformations. They are rapidly gaining popularity as they are flexible and in their limit can become universal function estimators [36, 133]. They are also quite popular since they are relatively easy to fit since the introduction of back propagation [127]; an efficient way to compute and store the gradients of a NN function with respect to its parameters using the chain rule of derivatives. Multiple Python libraries now offer tools to define and fit NNs. They can be used as prediction functions themselves or as building blocks in a more complex model. Let us now use a simple example to introduce NNs; to proceed we will begin with a simple fully connected single-layer NN, however, we also explain a few ways to generalize this simple model.

Figure 2.1 depicts a simple fully connected NN with one hidden layer ($\mathbf{z}$). In this instance, all inputs are linearly combined before going through a non-linear function σ called the activation function.

$$z_1 = \sigma\left((\sum_{j=1}^{m} \beta_{1,j} x_j) + b_1\right). \tag{2.10}$$

This process is done multiple time in parallel to allow for many different linear combinations to be learned, hence the multiple nodes z. Fixing $x_0 = 1$ allows to represent in a compact manner the transformation that occurs in the first layer

$$\mathbf{z} = \sigma\left(\mathbf{B}_1 \mathbf{x}\right), \tag{2.11}$$

where σ is applied element-wise and $\mathbf{B}$ is a matrix of parameters of size $q \times (m+1)$.

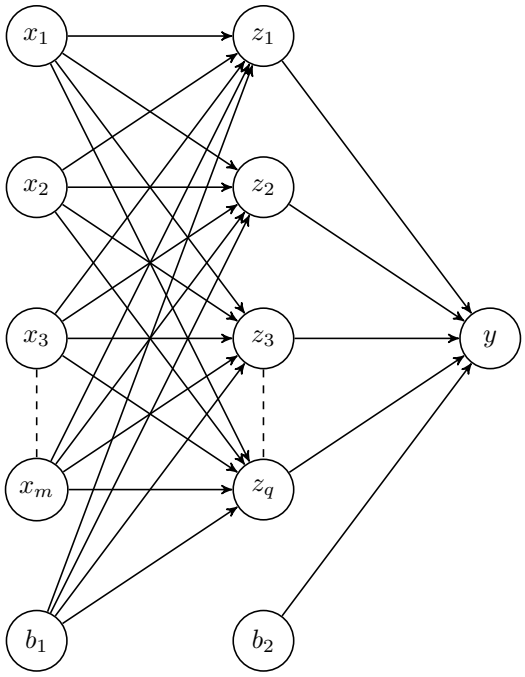


Figure 2.1: A simple NN with input of size m, 1 hidden layer of size q and a output layer of size 1.

In our simple model illustrated in Figure 2.1 the output y is of dimensions 1. Assuming a simple two-class classification problem, the logit activation function is used to serve as an estimator for p, the probability that $y = 1$ for instance,

$$y = \text{logit}\left(\mathbf{B}_2\sigma\left(\mathbf{B}_1\mathbf{x}\right)\right). \tag{2.12}$$

Usually, the number of hidden layers and the number of nodes in each of these layers are to be determined by the users. The number of output is usually determined by the problem itself. This is only a simple example so that we can visualize how NNs work, however there exist multiple way to adapt them to specific problems and there exist an extensive literature that covers these topics. For instance, one way to extend this simple model is to *add* hidden layers and sequentially repeat this process. This create a much more complex function and one that can grasp high order of interaction between predictors. They can also be generalized in other ways, for example CNNs [94] which does not rely on fully connected layers, we will see why in the next section.

Fitting the weight matrices $\mathbf{B}$ is done using back propagation [127]. Given a differentiable loss function and a differentiable activation functions σ we can compute the gradient of the loss function with respect to a single weight. Back-propagation is an efficient way to compute all of the needed gradients and since many of those gradients share some information in the chain derivative steps they must be all efficiently stored.

Using back propagation, NN's parameters can be estimated in any gradient based optimization problem. This is one of the reason NNs are so popular; they can be included as a building block in any model that is optimized using gradient-based techniques whereas DTs could not for instance. Since many ML techniques now rely on gradient-based optimizer, NNs became the go-to function estimator

in ML, replacing the linear combination in many models as NNs are more flexible and expressive than linear function while still being relatively easy to fit.

The expressive power of NNs has been studied extensively. Shalev-Shwartz and Ben-David [133] demonstrated that NNs are universal approximators for Boolean functions. That is rigorously, for every Lipschitz function $f : [-1, 1]^n \to [-1, 1]$ it is possible to construct a network such that for every input $\mathbf{x}$ the network outputs a number in $f(\mathbf{x}) + / - \varepsilon$. More recently, Csáji [36] claims in his thesis :

> The universal approximation theorem claims that the standard multilayer feed-forward networks with a single hidden layer that contains finite number of hidden neurons, and with arbitrary activation function are universal approximators in $C(R^m)$.

In the current thesis, NNs are central to any computer vision algorithms we studied, more precisely, CNNs which we introduce next. They are not only popular because of their flexibility and overall performances but also because of the large amount of support available. However there exist many interesting and important unanswered questions regarding NNs: how do you select the number of hidden layers or nodes in each layer for instance. These questions are slightly discussed in our research and are currently investigated in many research papers, however there are no established solutions yet. Nonetheless NNs are frequently used due to their empirical prowess.

2.1.5 Convolutional neural networks

In the computer vision field, a special NN structure has been widely used; Convolutional Neural Network (CNN) [94]. A CNN is well suited for image analysis as its architecture itself is designed to incorporate spatial correlation and some degree of shifts and scale invariance. LeCun et al. [95] identify three structural aspects of CNNs that ensure those properties: 1) local receptive fields, 2) shared weights and 3) spatial subsampling.

In a fully connected NN (Section 2.1.4), every input is passed through every node of the next layer. In the case of image analysis, this results in every pixel of an image being inputs of every function in the first layers. For one, this forces the NN to have a large number of parameters. It also neglects the correlation between nearby pixels; it considers pixels far from another potentially as correlated as the ones nearby. In CNNs this is usually taken care of by convolution layers. For these layers, only a small number of nearby pixels are passed as inputs to the next layer as illustrated in Figure 2.2 below.

These layers are said to have sparse connectivity which both reduces the memory requirements and increases the statistical efficiency of the model. It also means faster prediction as fewer operations are needed to emit a prediction. These layers also contribute towards parameter sharing in this model.

Another typical step in a CNN is pooling. A pooling function outputs a summary statistics of its inputs. For instance, the max pooling operation outputs the maximum of all the inputs. The mean input is another example of pooling function. These pooling stages are useful at making the representation invariant to small translation within the image.

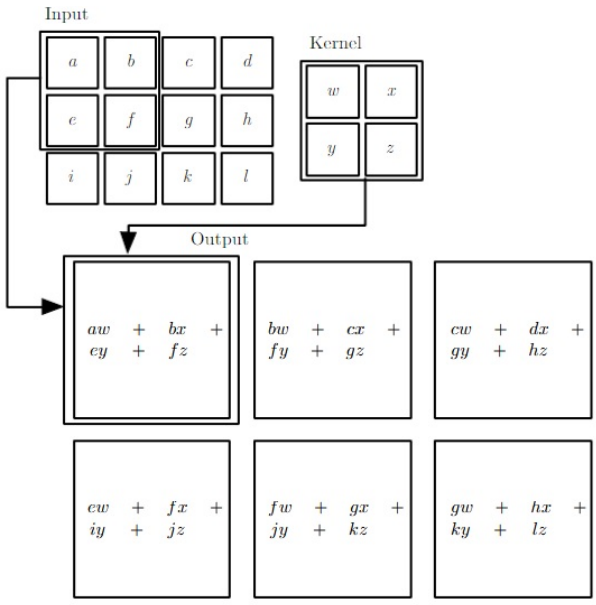


Figure 2.2: A visual representation of a convolutional layer included *Deep Learning* [55]

Usually a CNN contains multiple convolution layers, multiple pooling stages and fully connected layers. A detailed formulation of CNNs is available on Chapter 9 of *Deep Learning* by Goodfellow et al. [55].

We used CNNs in our latest research project presented in Chapter 7 as this is a computer vision project. CNNs are often used in this thesis as an example of special NNs; they've been built in order to tackle problems that are specific to computer vision and their performances are a proof that this algorithm design approach can be successful.

2.2 Unsupervised learning

In supervised learning we divide the variables into two groups; predictors and responses. Our goal is clear, establish or estimate the relationship between the predictors and responses, commonly formulated as predicting the responses using the predictors. However, which variables are responses and which one are predictors is usually chosen by the researcher, hence the *supervision* aspect. Under the assumption that $\mathbf{X} \times \mathbf{Y} \sim \mathcal{D}$, the supervised problem is to approximate $\mathcal{D}(\mathbf{Y}|\mathbf{X})$.

In unsupervised learning, no specific relationship between the variables is defined; variables are not labelled as predictor or response. Thus, we are trying to understand structures and patterns in the variables themselves. In other words, under the assumption that there exist a true data generating distribution $\mathcal{D}$, such that $\mathbf{X} \sim \mathcal{D}$ this time we are trying to estimate $\mathcal{D}(\mathbf{X})$ instead of a conditional distribution.

For example, clustering is a popular unsupervised procedure. Clustering is a statistical procedure that regroups data points that are alike. Another unsupervised task is compression where the task is to

find a lower-dimension representation of complex observations. In both of these tasks, all of the variables are equally considered and no label are assigned to any of the variables.

Another big difference between supervised and unsupervised learning is how the success of the model can be assessed. With supervised learning it is much easier to measure the success of an algorithm. The success being directly estimated by computing the averaged loss on unobserved data points, the test set. However, in the context of unsupervised learning there is usually no such obvious measure of success and this situation has led to multiple proposed methods where effectiveness is more a matter of opinion rather than rigorous computations.

2.2.1 Principal component analysis

Principal component analysis (PCA) is a common dimensionality reduction procedure in statistics. Even though we do not use it directly in any projects in this thesis we compare algorithms to PCA frequently and a brief introduction is warranted.

Suppose we have n observations of a continuous variable x, $S = \{\mathbf{x}_i : i \in (1, ..., n)\}$ of m dimensions. The purpose of PCA is to reduce the dimensionality of the data set by projecting data points to a much lower-dimensional space say d where $d << m$. In its simplest form the projection is done via a linear combination. Suppose z_i is the d-dimensional representation of x_i. The goal is to project S on a space that best captures the variability within the original space, thus we want to maximize the variance of the lower-dimensional representation.

To do so, let us define U as a $m \times d$ matrix called the projection matrix or compression matrix such that

$$\mathbf{z} = U^T \mathbf{x}.$$

The variance of the projected data becomes $U^T C U$ where C is the covariance matrix of the original data

$$C = \frac{1}{N} \sum_{n=1}^{N} (x_n - \bar{x})(x_n - \bar{x})^T.$$

To preserve rich information in the compressed space, we try to maximize the variance of the projected data, $U^T C U$. If we do so, we see the need for constrained on U and thus we set every vectors to be of length 1, i.e $\mathbf{u}^t \mathbf{u} = 1$ for every vector in U. In other words, theses vectors represent directions in the higher-dimensional space. To enforce this constraint we introduce a Lagrange multiplier which leads to the following maximization problem

$$U^T C U + \lambda(\mathbf{1} - U^T U).$$

Setting the derivative with respect to U equal to zero leads to

$$CU = \lambda U,$$

which says that U must be a matrix of eigenvectors of C. Finally, by left-multiplying by U^T we have

$$U^T C U = \lambda$$

and so the variance will be maximized if we set U as the collection of the eigenvectors of the data covariance matrix C associated with the d largest eigenvalues of C. Scaling can be an issue when using PCA and to circumvent this problem it is recommended to normalize S.

Algorithm 3 : PCA(S,d)
INPUT: A training set S and dimension of the lower representation d Start with the entire data set S 1 Compute the covariance matrix C of the data set S 2 Compute the matrix of eigenvectors U and the associated vector of eigenvalues λ 3 Extract the d eigenvectors U^d associated with d largest eigenvalues λ^d 4 Return U^d the projection matrix. OUTPUT: Projection matrix U^d.

2.2.2 Gaussian Mixture Model

Let us introduce the Gaussian Mixture Model. Moving from a simple Gaussian to a mixture of Gaussian allows for more complex observed-data distributions. The mixture component adds an extra layer of flexibility which allows observations to belong to one of many Normal distributions, thus enabling Gaussian models to account for multi-modal distributions among other things.

The simplest way to define such model is to consider z to be a latent discrete variable and $\mathbf{x}$ to be the observed continuous variable. From now on, we will refer to z as latent variables and we define latent variables as unobserved variables that have an effect on the observed variables. We will refer to $\mathbf{x}$ as the set of observed variable, usually the collected data.

For the Gaussian Mixture Model, z represents the component, i.e. if $z_i = k$ it means that x_i is distributed according to $N(\mu_k, \sigma_k)$. For every possible component k we have a distinct mean (μ) and standard deviation (σ), i.e. the parameters of the distribution of x depends on the latent component such that

$$p(x|z = k) = N(\mu_k, \sigma_k).$$

As we can see, the distribution for x is now a lot more complicated as we are only able to pin down the conditional distribution given z

$$\begin{aligned} p(x) &= \sum_z p(x, z) \\ &= \sum_{j=1}^{K} p(x|z = j)p(z = j), \end{aligned} \tag{2.13}$$

assuming we have K components. Since each z is associated with a different $p(x|z)$ we can see in Equation 2.13 why $p(x)$ is more expressive in this form. On the other hand, it is much harder to estimate the

parameters. If we attempt to compute the likelihood of an observed data set, applying the logarithm function is not enough to make the likelihood easily differentiable

$$\begin{aligned} p(x) &= \prod_{i=1}^{n} p(x_i) \\ &= \prod_{i=1}^{n} \sum_{j=1}^{K} p(x_i|z_i = j)p(z_i = j) \\ \Rightarrow \ln p(x) &= \sum_{i=1}^{n} \ln \sum_{j=1}^{K} p(x_i|z_i = j)p(z_i = j). \end{aligned} \tag{2.14}$$

Consequently, simple maximum likelihood strategies can not be employed here. The proposed solution to fit this model is the Expectation-Maximization (EM) algorithm. We will explain how this technique maximizes the likelihood function later. There exist intuitive formulations of the EM algorithm for GMMs but we introduce the Evidence Lower Bound - Kullback-Leibler (ELBO-KL) decomposition of the likelihood first. This is a bit of extra work, but it will prove useful when we introduce VAEs later.

The ELBO-KL decomposition

Let us demonstrate the ELBO-KL decomposition; a decomposition of the log likelihood of the observed variable in latent variable models. Notice that these equations hold for any distribution $q(z)$

$$\begin{aligned} \ln p(x) &= \ln \left(p(x,z)/p(z|x)\right) \\ &= \ln \left(p(x,z)\right) - \ln \left(p(z|x)\right) \\ &= \ln \left(p(x,z)\right) - \ln \left(p(z|x)\right) + \ln q(z) - \ln q(z) \\ &= \ln \left(\frac{p(x,z)}{q(z)}\right) - \ln \left(\frac{p(z|x)}{q(z)}\right) \\ \Rightarrow \mathbf{E}_{q(z)}[\ln p(x|\theta)] &= \mathbf{E}_{q(z)}\left[\ln \left(\frac{p(x,z)}{q(z)}\right)\right] - \mathbf{E}_{q(z)}\left[\ln \left(\frac{p(z|x)}{q(z)}\right)\right] \\ \Rightarrow \ln p(x|\theta) &= \mathbf{E}_{q(z)}\left[\ln \left(\frac{p(x|z)p(z)}{q(z)}\right)\right] - \mathbf{E}_{q(z)}\left[\ln \left(\frac{p(z|x)}{q(z)}\right)\right] \\ &= \mathcal{L}(q,p) + KL(q||p). \end{aligned} \tag{2.15}$$

Notice that since the KL divergence is greater or equal than 0, then $\mathcal{L}(q,p)$ is a lower bound for the likelihood. It is called the evidence lower bound (ELBO) or the variational lower bound. Multiple techniques for inference on graphical models are based upon the maximisation of this lower bound.

The Jensen's inequality is another way to demonstrate why the ELBO is a lower bound

$$\begin{aligned} p(x) &= \int p(x|z)p(z)dz \\ &= \int p(x|z)p(z)\frac{q(z)}{q(z)}dz \\ &= \int \frac{p(x|z)p(z)}{q(z)}q(z)dz \end{aligned} \tag{2.16}$$

$$\Rightarrow \ln p(x) \geq \int \ln \frac{p(x|z)p(z)}{q(z)} q(z)dz$$
$$= \mathbf{E}_{q(z)} \left[\ln \left(\frac{p(x|z)p(z)}{q(z)} \right) \right].$$

The EM algorithm : Maximization of the variational lower bound for tractable posterior

The Expectation-Maximization algorithm is an iterative procedure that slowly increases the value of the variational lower bound with two distinct steps. To begin, we will explain the key ideas and how the algorithm intends to maximize the likelihood and then we will see how this procedure leads to a maximization of the variational lower bound.

We've already discussed the issue of maximizing the likelihood of the observed data set in models with latent variables. Remember that

$$\ln p_\theta(x) = \sum_{i=1}^{n} \ln \sum_{j=1}^{K} p_\theta(x_i|z_i = j)p_\theta(z_i = j),$$

and thus maximizing the likelihood is analytically impossible. This model contains $(k-1)+2k$ parameters. The $k-1$ parameters for the categorical distribution of z, the k means μ_j and the k variances σ_j^2 that correspond to the k normal components. From now on, we refer to all of these parameters as θ.

Since we only observe $\mathbf{x}$, the only information we have about $\mathbf{z}$ is through the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. Therefore we cannot directly use the complete-data log likelihood $\ln p_\theta(\mathbf{x}, \mathbf{z})$ and instead we will compute the expectation of the complete log likelihood under the posterior distribution with the current set of parameter estimates

$$\mathbf{E}_{p_{\theta^o}(\mathbf{z}|\mathbf{x})}[\ln p_\theta(\mathbf{x}, \mathbf{z})] = \sum_{\mathbf{z}} p_{\theta^o}(\mathbf{z}|\mathbf{x}) \ln p_\theta(\mathbf{x}, \mathbf{z}) = \mathcal{Q}(\theta, \theta^o). \tag{2.17}$$

Computing this expectation is the **E** step of the EM algorithm. Then,the **M** step is the maximization of $\mathcal{Q}(\theta, \theta^o)$ with respect to θ. Here, θ^o stands for old θ and is the set of parameters under which we computed the posterior $p_{\theta^o}(\mathbf{z}|\mathbf{x})$. In some cases, the **M** step can be done analytically.

Specifically for a mixture of Gaussian, both steps are quite simple. The posterior $p_{\theta^o}(\mathbf{z}|\mathbf{x})$ is

$$p_{\theta^o}(z_n = k|\mathbf{x}_n) = \frac{\pi_k N_{\theta_k}(\mathbf{x}_n)}{\sum_{j=1}^{K} \pi_j N\theta_k(\mathbf{x}_n)} = \gamma(z_k), \tag{2.18}$$

where $\pi_k = p(z = k)$. Then optimizing $\mathcal{Q}(\theta, \theta^o)$ leads to the following estimates

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\mathbf{x}_n$$
$$\Sigma^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \mu_k^{new})(\mathbf{x}_n - \mu_k^{new})^T$$
$$\pi_k^{new} = \frac{N_k}{N},$$

where $N_k = \sum_{n=1}^{N} \gamma(z_{nk})$. Full details and proofs are available in Hastie et al. book [61].

However, we are going to use the ELBO-KL decomposition to demonstrate how the EM algorithm succeed at maximizing the likelihood and we will motivate the need for other techniques.

We previously demonstrated that $\mathcal{L}(q_\varphi, p_\theta)$ is a lower bound for the log-likelihood of the observed data $\ln p_\theta(\mathbf{x})$. The ELBO is function of the parameters θ of the distribution p_θ and of the parameters φ of a distribution over the latent variables $q_\varphi(\mathbf{z})$. Let's demonstrate how both steps of the EM algorithm increase $\mathcal{L}(q_\varphi, p_\theta)$ in their own way. In the **E** step, we maximize $\mathcal{L}(q_\varphi, p_\theta)$ with respect to φ while in the **M**, we then maximize $\mathcal{L}(q_\varphi, p_\theta)$ with respect to θ.

The **E** step considers the effect of $\mathbf{z}$ through the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$ under the current set of parameters, and then compute the expectation of the complete log-likelihood under that posterior distribution. The **E** step consist of setting $q_\varphi(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$. We know that $\mathcal{L}(q_\varphi, p_\theta) = \ln p_\theta(\mathbf{x}) - KL(q||p)$ and thus in this formulation the ELBO depends on φ only in the KL component. Setting $q_\varphi(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x})$ makes the KL divergence vanish which effectively maximize $\mathcal{L}(q_\varphi, p_\theta)$. This also highlights one of the main assumptions necessary to use an EM algorithm, we need to be able to compute $p_\theta(\mathbf{z}|\mathbf{x})$.

In the following **M** step, we maximize $\mathcal{Q}(\theta, \theta^o)$ with respect to the parameters θ. Let's actually see what happens when we substitute $q_\varphi(\mathbf{z})$ by $p_{\theta^o}(\mathbf{z}|\mathbf{x})$ in the lower bound

$$
\begin{aligned}
\mathcal{L}(q_\varphi, p_\theta) &= \sum_{\mathbf{z}} q_\varphi(\mathbf{z}) \ln \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\varphi(\mathbf{z})} \right) \\
\Rightarrow \mathcal{L}(p_{\theta^o}(\mathbf{z}|\mathbf{x}), \theta) &= \sum_{\mathbf{z}} p_{\theta^o}(\mathbf{z}|\mathbf{x}) \ln \left(\frac{p_\theta(\mathbf{x}, \mathbf{z})}{p_{\theta^o}(\mathbf{z}|\mathbf{x})} \right) \\
&= \sum_{\mathbf{z}} p_{\theta^o}(\mathbf{z}|\mathbf{x}) \ln p_\theta(\mathbf{x}, \mathbf{z}) - \sum_{\mathbf{Z}} p_{\theta^o}(\mathbf{z}|\mathbf{x})) \ln p_{\theta^o}(\mathbf{z}|\mathbf{x})) \\
&= \mathcal{Q}(\theta, \theta^o) + \text{const.}
\end{aligned}
$$

In the **M** step, we maximize $\mathcal{Q}(\theta, \theta^o)$ with respect to θ. This maximizes the ELBO in parallel since the two are equal up to a constant.

Assuming the EM algorithm is successful at fitting such latent variable models it is reasonable to ask why would we need any other techniques to optimize the parameters in latent variable models. The problem with EM is that it requires us to compute $p_{\theta^o}(\mathbf{z}|\mathbf{x})$ which may not be feasible in some cases. The proposed solution in those cases is to directly maximise the ELBO in another way.

In this thesis, GMM and PCA often serve the role as a reference point when discussing more complicated latent variable models, they are widely used and well-known and so are their limitations.

2.2.3 Variational autoencoders

Before discussing the Variational AutoEncoder (VAE) model, let us introduce the simple AutoEncoder (AE) as a generalized PCA algorithm.

An AE is a model that simultaneously learn how to compress and decompress data. More generally, it is possible to train AEs to project data to a higher dimensional space but here we will assume the latent representation is of lower dimension.

Assuming we have n observations $S = \{\mathbf{x}_i : i \in (1, ..., n)\}$ of m dimensions. We want to learn a simple deterministic function q that projects the data to a lower dimension representation of size d, i.e. $q : \mathcal{X} \to \mathcal{Z}$, where $\mathcal{X}$ is the observed data space of size m and $\mathcal{Z}$ is the latent representation space of size d. Simultaneously, we learn a decompression function p from the projected space (d) to the full dimension (m) space $p : \mathcal{Z} \to \mathcal{X}$. An AE simultaneously learns q and p usually by minimizing some metric of reconstruction error.

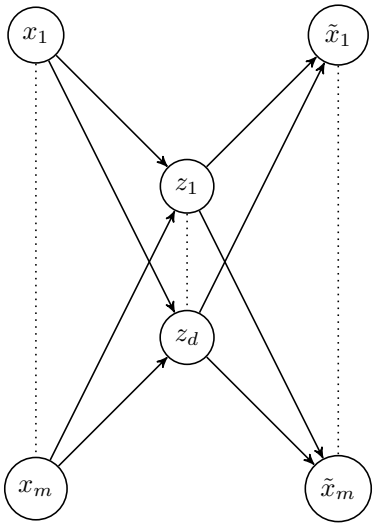


Figure 2.3: A simple AE where $\tilde{x}$ is the reconstructed x, i.e. $p(q(x)) = \tilde{x}$.

Under the assumptions that both q and p are simple linear combinations and that we minimize the mean squared reconstruction error $\sum_{i=1}^{n} ||p(q(x_n)) - x_n||^2$ the solutions of such system are given by the d eigenvectors associated with the d biggest eigenvalues; the PCA solution. Proofs are available in Bishop's book [15]. We previously introduced PCA as a solution to the maximum variance lower-dimensional projection problem but it turns out it is also the solution for this minimal reconstruction error problem. Since PCA is well-know in the statistical community and AEs are well-know in the machine learning community, this results create a nice intuitive way to tie together these two algorithms.

However, an AE can be generalized in many ways: it can be made of non-linear functions for example and those functions can be optimized with respect to another objective function.

VAEs [86, 83] form a flexible family of latent variable models. Two major changes to the classic AE are made: (1) we assume a distribution on the latent variable z and a distribution on the observed variable x which results in a probabilistic model and (2) p and q are NNs. Because we have a probabilistic model, p maps the latent variable to the parameters of the observed distribution and similarly q maps the observed variable to the parameters of the latent distribution.

For GMMs, we have such probabilistic mapping; the component z affect the parameters of the observed normal distribution. In this case, the mapping from z to x is $\theta : \{1, .., k\} \to (\mathbb{R}^m) \times (\mathbb{R}^m \times$

$\mathbb{R}^m$). Each component k has it's own set of parameters: if $z = j$ then $p(x|z) = N(\mu_j, \sigma_j)$. For our demonstration of VAEs, we assume the distribution of x is still Gaussian but we also assume that z is Gaussian as well. Typically, it is assumed that $z \sim N(0, I)$. Since z is now continuous, we could identify this model as a Gaussian mixture where we have infinitely many components. Furthermore, to allow this, the parameters are now a continuous function of the latent variable z; $\theta = [\mu_x, \sigma_x] = f_x(z)$. To use a short notation, we identify $\mu_x(z)$ as the function that takes z as input and return the parameters μ_x associated with this value and same for $\sigma_x(z)$ or $\theta(z) : \mathbb{R}^d \to \mathbb{R}^m \times \mathbb{R}^m_+$ because x are independent given z in most implementations. Because θ is a continuous function, it ensures that points that are alike (close to one another) in the latent space are also near in the observation space.

We define z as the latent representation of the observation x or as its code and $\theta(z)$ as the decoding function which takes in the code and return the parameters of the observed-data distribution. We use a neural network (NN) function as decoding function. It has the benefit of allowing for a maximum amount of flexibility but in turn makes the posterior of the latent $p_\theta(z|x)$ intractable analytically and consequently the EM algorithm cannot be used.

The proposed solution is to estimate $p_\theta(z|x)$ with an approximate distribution $q_\varphi(z|x)$. This technique is know as variational inference and we call the approximate distribution $q_\varphi(z|x)$ the variational distribution. With that variational distribution, the resulting ELBO is

$$\mathcal{L}(\varphi, \theta) = \mathbf{E}_{q_\varphi(z|x)} \left[\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)\right]$$

We define $q_\varphi(z|x)$ as the encoding distribution. The effect of x on the encoding distribution is reflected in the parameters φ. We assume q_φ to be a normal distribution and once again we explain the parameters φ as a function of x; $\varphi = [\mu_z, \sigma_z] = f_z(x)$ or $\varphi(x) : \mathbb{R}^m \to \mathbb{R}^d \times \mathbb{R}^d_+$. This function is set to be a NN that we will refer as $\varphi(x)$ or $\mu_z(x)$ and $\sigma_z(x)$ from now on.

Since we cannot directly compute $\mathbf{E}_{q_\varphi(z|x)} \left[\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)\right]$ we will sample Monte Carlo estimates of the ELBO. In other words, we draw z from $q_\varphi(z|x)$ and then compute $\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)$ for a batch of observations. After, we maximize the batch ELBO Monte Carlo estimate with respect to the parameters of both NN functions, $\theta(z)$ and $\varphi(x)$. This allows us to train a VAE.

Algorithm 4 : Optimize VAE(x,d)

INPUT: training observations x, size of the latent representation d
1) Process observations x through the NNs φ to produce $\varphi(x)$
2) Sample z from $q_{\varphi(x)}(z|x)$
3) Process latent sample z through the NNs θ to produce $\theta(z)$.
4) Compute $\ln p_{\theta(x)}(z) + \ln p_{\theta(z)}(x|z) - \ln q_\varphi(z|x)$, the ELBO Monte Carlo estimate.
5) Maximize the ELBO with respect to the weights of φ and σ using back propagation
Repeat 1-5 until convergence.
OUTPUT: Trained VAE; weights of the NNs φ and θ optimized to maximize the ELBO.

VAEs are central to the last three research projects contained in this thesis. First we adapt VAEs for survival analysis, a very well-know type of data analysis in statistics, this is discussed in Chapter

6. Second, most of our work in computer vision is built on VAE: for instance our experiments with controllable generative model rellied on VAEs as explained in Chapter 7. Finally, we noticed a large gap between the theory of VAE and common implementations. These differences are mostly unacknowledged in the literature and this is a topic we address in Chapter 5.

Chapter 3

Analysis of an academic data set

An application of random forests

In this chapter, a large data set containing every course taken by every undergraduate student in a major university in Canada over 10 years is analysed. To begin, the first two semesters of courses completed by a student are used to predict if they will obtain an undergraduate degree. Secondly, for the students that completed a program, their major is predicted using once again the first few courses they have registered to. We selected random forests as classifiers for this project because these models are not built on any assumptions about the data distributions and are easy to use *out of the box*. They also allow for reliable variable importance measurements. These measures explain what variables are useful to the classifiers and can be used to better understand what is statistically related to the students' situation.

In this chapter, we demonstrate (1) the potential of Random Forest models as simple and easy-to-use alternatives to linear models and (2) the strength of variable importance analysis to obtain useful information difficult to gather with traditional statistical models. The main contributions of this chapter were introduced first in our article *Predicting University Students' Academic Success and Major using Random Forests* [9] published in *Research in Higher Education*.

3.1 Introduction

Being able to predict if a student is at risk of not completing its program is valuable for universities that would like to intervene and help those students move forward. Predicting the major that will be completed by students is also important in order to understand as soon as possible which program attracts more students and allocate resources accordingly. Since gathering data can be an expensive procedure, it would be useful being able to predict both of these things using data the university already possesses such as student records. Understanding which variables are useful in both of these predictions is important as it might help understand what drives student in taking specific classes.

These two prediction problems are classification problems. To solve these, a popular machine learning algorithm is used, a Random Forest (RF). A RF is a collection of classification trees which naturally

allows interactions of high degree across predictors. The RF uses the first few courses attempted and grades obtained by students in order to classify them. A RF can also be used to assess variable importance in a reliable manner.

The University of Toronto provided a large data set containing individual-level student grades for all undergraduate students enrolled at the Faculty of Arts and Science at the University of Toronto - St. George campus between 2000 and 2010. The data set contains over 1 600 000 grades and over 65 000 students. This data set was studied by Bailey et al. [6] and was used to build an adjusted GPA that considers course difficulty levels. Here, RF classifiers are built upon this data set and these classifiers are later tested.

The contribution in this chapter is two-fold. First we demonstrate the high performance of RF as *out-of-the-box* model which obtains higher prediction accuracy than linear classifiers thus making them useful for universities that would like to predict where their resources need to be allocated. Second, the variable importance analysis produced by the RF contains lots of information. Among many things, the high importance of grades in low-grading departments was noted; this is often identified as a symptom of grade inflation in higher education.

3.2 Literature review

3.2.1 Predicting success

In this chapter a statistical learning model is established to predict if a student succeeds at completing an undergraduate program and to predict what major was completed. This statistical analysis of a higher education data set shares similarities with recent articles by Chen and Desjardins [27, 28] and Leeds and DesJardins [96] as a statistical approach is introduced, a data set is presented and policy making implications is discussed. The task of predicting student academic success has already been undertaken by many researchers. Recently Kappe and van des Flier [78] tried to predict academic success using personality traits. In the meanwhile, Glaesser and Cooper [54] were interested in the role of parents' education, gender and other socio-economic metrics in predicting high school success.

While the articles mentioned above use socio-economic status and personality traits to predict academic success, many researchers are looking at academic-related metrics to predict graduation rates. Johnson and Stage [75] use High-Impact Practices, such as undergraduate research, freshman seminars, internships and collaborative assignments to predict academic success. Using regression models, they noted that freshman seminars and internships were significant predictors. Niessen and al. [112] discuss the significance of trial-studying test in predicting student dropouts. This test was designed to simulate a representative first-year course and student would take it before admission. The authors noted that this test was consistently the best academic achievement predictor.

More recently, Aulck and al. [5] used various machine learning methods to analyse a rather large data set containing both socio-economic and academic metrics to predict dropouts. They noted similar performances for the three methods compared; logistic regression, k-nearest neighbours and random forests. The proposed analysis differs from the above-mentioned as it takes on the challenge to predict academic success and major using strictly academic information available in student records. The benefits

of having classifiers built upon data they already own is tangible for university administrations. It means universities would not need to force students to take entry tests or rely on outside firms in order to predict success rate and major which is useful in order to prevent dropout or to allocate resources among departments. As noted by Aulck and al. [5] machine learning analysis of academic data has potential and the uses of random forest in this chapter aims at exploiting this potential.

3.2.2 Identifying important predictors

Identifying and interpreting the variables that are useful to those predictions are important problems as well. It can provide university administrators with interesting information. The precise effect of grades on a student motivation lead to many debates and publications over the years (more recently [109, 113]). Because grades should be indicators of a student's abilities, evaluating the predictive power of grades in various departments is important. University administrators might want to know if grades in a department are better predictors than grades in other departments. Continuing on the point, it is also important to understand what makes the evaluations in a department a better indicator of students' success. Random forest mechanisms lead to variable importance assessment techniques that will be useful to understand the predictive power of grades variables.

Understanding the importance ranking of grades in various departments can also enlighten us regarding the phenomenon of *grade inflation*. This problem and some of its effect has been already discussed in many papers ([130, 76, 7]) and it is consensual that this inflation differs from one department to another. According to Sabot and Wakeman-Linn, [130] this is problematic since grades serve as incentives for course choices for students and now those incentives are distorted by the grade inflation. As a consequence of the different growths in grades, they noted that in many universities there exist a chasm in grading policies creating high-grading departments and low-grading departments. Economics, Chemistry and Mathematics are examples of low-grading departments while English, Philosophy and Political Science are considered high-grading.

As Johnson mentions [76], students are aware of these differences in grading, openly discuss them and this may affect the courses they select. This inconsistency in course difficulty is also considered by Bailey and al. [6] as they built an adjusted GPA that considers course difficulty levels. The accuracy of that adjusted GPA in predicting uniform test result is a great demonstration that courses do vary in difficulty. If some departments suffer from grade inflation, the grades assigned in that department should be less tied to the actual student ability and therefore they should be less predictive of student success. A thorough variable importance analysis will be performed in order to test this assumption.

Understanding which predictors are important can also provide university administrators with feedback about some of their programs. For example, some of the High-Impact Practices identified by Johnson and Stage [75] are part of the University of Toronto's program. The variable importance analysis could be a useful tool to evaluate the effect of such practices.

3.3 Methodology

3.3.1 Data

The data set provided by the University of Toronto contains 1 656 977 data points, where each observation represents the grade of one student in one course. The data was collected over 10 years. A data point is a 7 dimensions observation containing the student ID, the course title, the department of the course, the semester, the credit value of the course and finally the numerical grade obtained by the student.

Since this is the only data obtained, some pre-processing is required in order for algorithms to be trained. The **first research question** is whether it is possible to design an algorithm which accurately predicts whether or not a student will complete their program. The **second research question** is whether it is possible to design an algorithm which accurately predicts, for students who complete their program, which major they will complete. These two predictions will be based upon first-year student records.

The data has been pre-processed for the needs of the analyses. At the University of Toronto, a student must complete 20 credits in order to obtain an Honours B.A. or B.Sc [148]. A student must also either complete 1 Specialist, 2 Majors or 1 Major and 2 Minors. The first five credits attempted by a student roughly represent one year of courses. Therefore, for each student every semester until the student reaches 5 attempted credits are used for prediction. It means that for some students, the predictors represent exactly 5 attempted credits and for some other students, a bit more. The set of predictors consists of the number of credits a student attempted in every department and the average grade across all courses taken by the student in each department. Since courses were taken by students in 71 different departments, the predictor vector is of length 142. Of course, many other predictors could also be computed from the data set, but these are the most appropriate ones for the purpose of the variable importance analysis.

The missing values for grades were replaced by values outside of the domain, -1 in this case. To answer the first research question, a binary response indicating whether or not a student completed their program is needed. Students that completed 18 credits were labelled as students who completed their program. Students who registered to 5 credits worth of courses, succeeded at fewer than 18 credits worth of courses and stopped taking courses for 3 consecutive semesters are considered students who began a program but did not complete it. Since some students take classes in other faculties or universities, 18 credits was deemed a reasonable threshold. It is possible that some students did not complete their program even though they completed 18 credits, but it is more likely that they took courses in other faculties or universities. To be considered dropouts, only students who registered to at least 5 credits worth of courses were considered. It was assumed that students that registered to fewer credits were registered in another faculty, campus, university or were simply auditing students. After this pre-processing was performed, the data set contains 38 842 students of which 26 488 completed an undergraduate program and 12 294 did not.

Additionally, since the data set was collected over 10 years but it takes 4 years to complete an undergraduate program plenty of cohorts were given the opportunity to complete their program within the time frame the data was collected. Nevertheless, there might be some bias in the records included

because slow students might be under-represented. We assume this bias to be negligible for our analysis; we assume that the speed at which student complete their program has a negligible effect on both responses.

To answer the second research question a categorical response representing the major completed by the student is required. To do so, the 26 448 students who completed a program are kept. The response will represent the major completed by the student. Since this information is not available in the data set, the department in which the student completed the largest number of credits is considered the program they majored in. Therefore, the response variable is a categorical variable that can take 71 possible values. This formatting choice might be a problem for students who completed more than 1 major. Some recommendations to fix that problem can be found in the conclusion.

Regarding the various grading policies of this university it was noticed that Mathematics, Chemistry and Economics are the three departments with the lowest average grades. As grades do vary widely across the data set there is no statistically significant difference between the departments but we did observe nonetheless that departments that were defined as low-grading departments in many papers do appear as the lowest grading departments in this data set too.

3.3.2 Techniques

The analysis is performed using Random Forests as introduced in Chapter 2. In order to train our model, set hyper-parameters and test our model, we divide the data set in three parts. The algorithm is trained upon the training set, which contains 90% of the observations in order to learn from a large portion of the data set. 5% of the data set is assigned to the validation set which is utilized to select various optimization parameters. Finally, the rest of the data set is assigned to the test set, which is a data set totally left aside during training and later used to test the performances of the trained classifier.

In this project, random forests were used because they are stable classifiers with many interesting properties. Let us explain why RFs would be prefer over the logistic regression. To begin, RF are built upon very few data assumptions; in this case predictors are collinear which would be a problem for the logistic regression. We also assume there exist interactions between the predictors, given we have 142 predictors, including interactions of high degree in a logistic regression would be difficult. Finally, RFs also allow us to compute the variable importance which evaluates the importance of individual predictors throughout the entire prediction process. As discussed in Chapter 2, this is something Logistic regression cannot provide. Let us further discuss variable importance techniques in RFs.

3.3.3 Variable Importance in Random Forests

A variable importance analysis aims at understanding the effect of individual predictors on the classifier output. A predictor with a great effect is considered an important predictor. The ability to assess the importance of a predictor is something neither the p-value nor the size of the coefficient is suited to do. A RF provides multiple variable importance computations. The *Gini decrease importance* sums the total impurity measure decrease caused by partitioning upon a predictor throughout an entire tree and then computes the average of this measure across all trees in a forest. This technique is tightly related to the construction process of the tree itself and is pretty easy to obtain as it is non-demanding computationally.

As explained in Chapter 2, the *permutation decrease importance* computes the importance of a predictor for an algorithm based on how much the accuracy of the given algorithm drops when the predictor is permuted. If a predictor has a significant effect on the response, the algorithm should lose a lot of prediction accuracy if the values of that predictor are permuted in the test set.

Storbl & al. [139] recently published an article where these techniques are analysed and compared. According to this paper, the selection bias of the decision tree procedure might lead to misleading variable importance. Numerous papers [22, 82, 89] noticed a selection bias within the decision tree procedure when the predictors are of different nature. The simulation studies produced by Storbl & al. [139] show that the Gini decrease importance is not a reliable variable importance measure when predictors are of varying types. The Gini decrease importance measure tends to overestimate the importance of continuous variables.

It is also shown [139] that the variable importance techniques described above can give misleading results due the replacements when drawing bootstrap samples. It is recommended that researchers build RFs with bootstrap samples without replacements (a random subsample) and use an unbiased tree-building procedure [103, 82, 102, 66]. If a classic tree-building procedure is used, predictors should be of the same type or only the permutation decrease importance is reliable. We respected those recommendations when we computed the variable importance.

3.3.4 Algorithms

A classification tree using the Gini impurity as splitting criteria was coded in the C++ language using the Rcpp library [43]. The algorithm proceeds as explained in Chapter 2, the tree it produces is unpruned and training sets are partitioned until they contain only 50 observations. Three versions of the RF algorithm are going to be used. Even though one of these models outperforms to two others in terms of prediction accuracy, the variable importance analysis of all three models are considered and aggregated. For clarity and conciseness purposes, only the best model's performance is included but performances across the three forests are quite similar. **Random forest # 1** consists of 200 trees and can split upon every variable in each region. Bootstrap samples are drawn without replacement and contain 63% of the original training set. **Random forest # 2** fits 200 trees but randomly selects the variable to be partitioned upon in each region.

Finally, the popular R RandomForest package [97] was also used. It is an easy to use and reliable package that can fit RFs and produce variable importance plots. Using this package, **random forest # 3** was built. It contains 200 trees. Once again, bootstrap samples are drawn without replacement and contain about 63% of the size of the original training set. By default, this algorithm randomly selects a subset of inputs for each region. Regarding the impurity measure, the Gini impurity was selected because it has important theoretical properties, such as being differentiable, and has been performing well empirically.

Linear models were trained for both of the classification problems serving as benchmarks. In order for the comparison to be as direct as possible, the linear model classifiers were constructed upon the same set of predictors; it may be possible to improve both the RF and the linear model with different predictors. As the problems are two classification ones, the linear model selected was a logistic model for the first research questions and a multinomial logistic model for the second research question.

3.4 Results

3.4.1 First research question : Predicting program completion

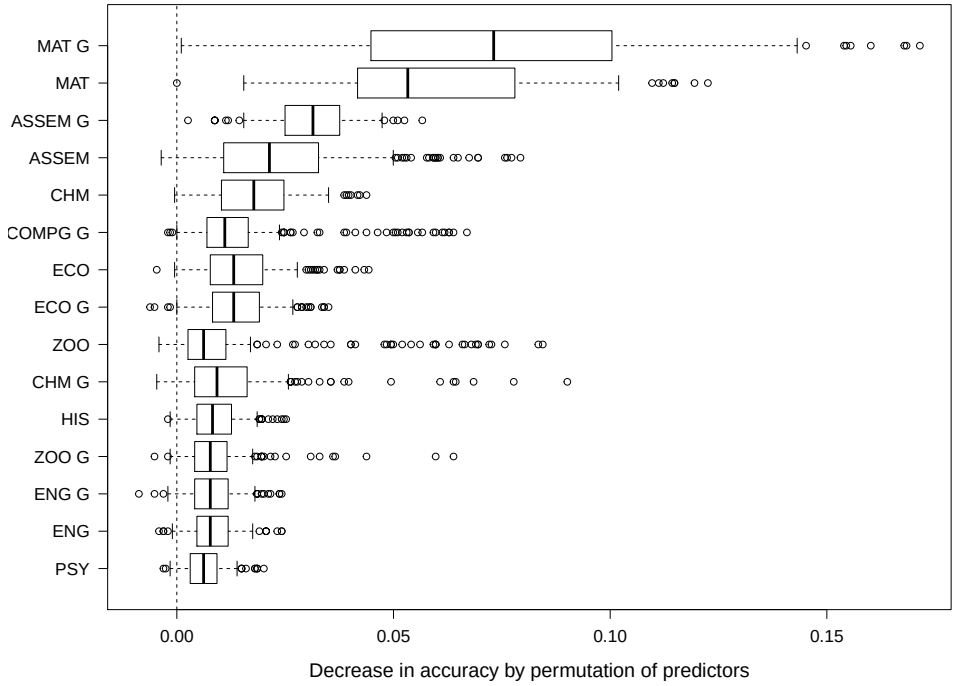
Random forest # 3 produced the best accuracy on the test set. Among the students who completed their program in the test set, the classifier achieves 91.19% accuracy. Out of the 418 students who did not complete their program, the classifier achieves 52.95% accuracy. The combined result is 78.84% accuracy over the complete test set.

This is higher accuracy than if all students would be classified as students who competed their program, a very naive classification technique, which would result in 68.08% accuracy. The RF accuracy is also higher than the 74.21% accuracy achieved with a logistic regression fit on the same predictors. These predictions can be useful for university administrations that would like to predict the number of second-year students and prepare accordingly with a sufficient margin. About 75% of students identified as dropouts by the RF classifier are true dropouts. Therefore students identified as dropouts by the algorithm could be considered higher-risk students and these predictions could be useful in order to target students in need of more support to succeed. The relatively high accuracy of the classifier is also an indicator that the variable importance analysis is reliable.

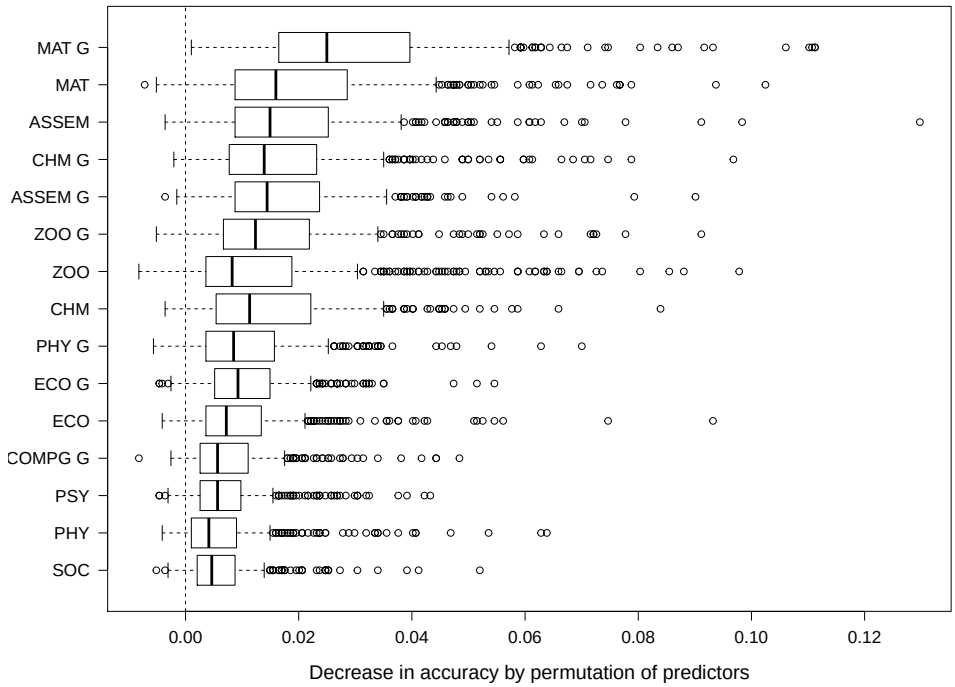
Variable importance is determined by the average decrease in accuracy in the test set caused by a random permutation of the predictor. This technique has been selected since it is more reliable as explained in Section 3.3.3. The top 15 variables according to the permutation decrease were kept and ordered in Figure 3.1. Since variable importance varies from one model to another, the three variable importance plots were included and the results will be aggregated.

Since these RFs are meant to predict student's success at completing an undergraduate program, we think it is interesting to observe which predictor is important in producing such prediction. In Figure 3.1 and for all the following figures, the variable representing the number of credits in a department is identified by the department code, i.e. the number of credits in Chemistry is identified by CHM. The variable representing the averaged grade in a department is identified by the department code followed by the letter G, i.e CHM G represents the averaged grade in Chemistry.

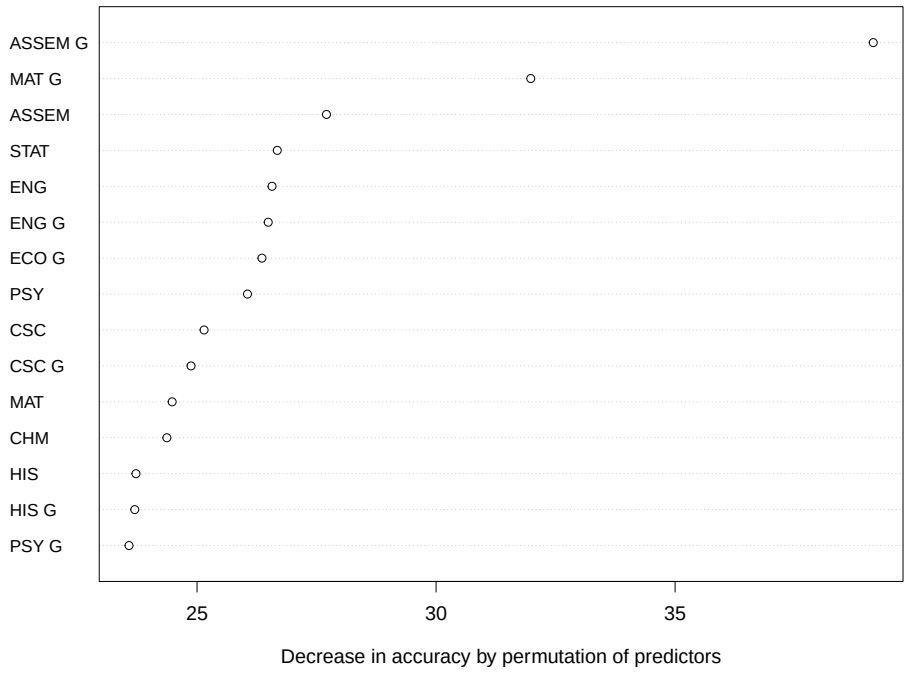
To begin, it was also noted that the variance for the grade variables were larger. Across all three RFs, the grades in Mathematics (MAT), Finance (COMPG), Economics (ECO) are consistently among the most important grade variable. These departments are considered low-grading departments and perhaps the strict marking of these departments helps to better distinguish students among themselves and successfully predict their program completion status. A possible explanation is that the grade inflation that suffered the high-grading departments caused the grades in those departments to be no longer a reliable tool estimate student students abilities which could be a symptom of grade inflation as suggested in Section 3.2.2. Other factors could have caused this phenomenon such as less sequential courses in Human Science fields, larger classes size, reduced access to a professor or other factors. Regardless, we noticed how grades in different departments contribute much more to the model than grades in other departments. Therefore, universities could use such technique to verify if grades in a department have more predictive power than grades in other departments and act accordingly since grades are meant to represent students' abilities.



(a) **Random forest # 1**.



(b) **Random forest # 2**.



(c) **Random forest # 3**.

Figure 3.1: Permutation decreases importance boxplots for the first research question.

We also noticed the importance of ASSEM in the three variable importance plots. The ASSEM code represents a special type of first year seminar course. It seems that the students that registers in theses courses are easy to classify as both grades and the number of credits are considered important. This result agrees with the result obtained by Johnson and Stage [75] about the importance of first year seminar courses. The first year seminar courses (ASSEM) were brand new at the University of Toronto and the analysis performed provided evidence of the merit of such courses in order to establish a student's profile and to predict success. In other words, such variable importance analysis could help university administrations assess the usefulness of new programs and courses.

3.4.2 Second research question : Predicting the major

The second task at hand is to build a RF that predicts the student's major. Once again, from a prediction accuracy perspective, **random forest # 3** offered better performances with 47.41% accuracy when predicting the major completed. This appears slightly lower than expected, but considering there are 71 different programs, being able to pin down the right program for about half of the students seems successful. This is a better result than the meager 4.75% obtained by assigning majors with probabilities weighted by the proportion of the majors completed. The 47.41% accuracy of the RF is also significantly better than the 42.63% accuracy obtained with the multinomial logistic regression benchmark. For classification purposes, these classifiers could help individual departments predict the number of students registering to second, third or fourth year courses and graduate programs. Predicting the major could also help university administrations to allocate the financial resources among the departments or to

decide the programs that require more advertisements.

Though harder to interpret, variable importance can also provide additional information with respect to that research questions. Here is the variable importance analyses produced by the three RFs; once again, the 15 most important predictors are displayed. The importance of a predictor is determined by the average decrease in accuracy in the test set caused by a random permutation of the predictor.

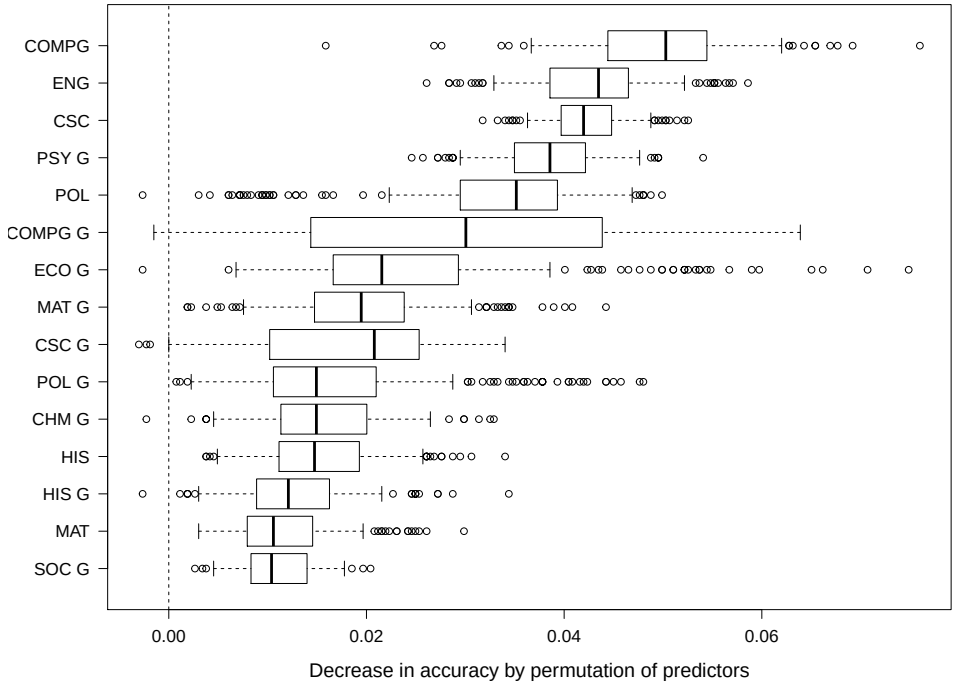
A decrease in importance for the grades variable is noted in Figure 3.2. This was to be expected because of how the data was formatted. Since the department in which the highest amount of credit was obtained is considered the major completed by the student, these variable importance measures are not surprising. Actually, if all the courses were included, instead of only the first year, the amount of credit in every department precisely defines the response variable. Considering this weakness in the data formatting, the grades still have a relatively high importance. It seems hard to see any effect of grading policies in the predictive power of grades regarding that research question.

It seems like for some departments, such as English (ENG) and Computers Sciences (CSC), it is easy to predict students that will complete a major in those departments by almost solely looking at the number of courses attempted in those departments during the first year. This is caused by the fact that a vast majority of students that take courses in Computers Science or English during their first year end up completing an undergraduate program in these departments respectively. From a policy-making perspective, departments could use this information as they might want to adapt the content of their first-year courses now that they know more about the audience of these courses.

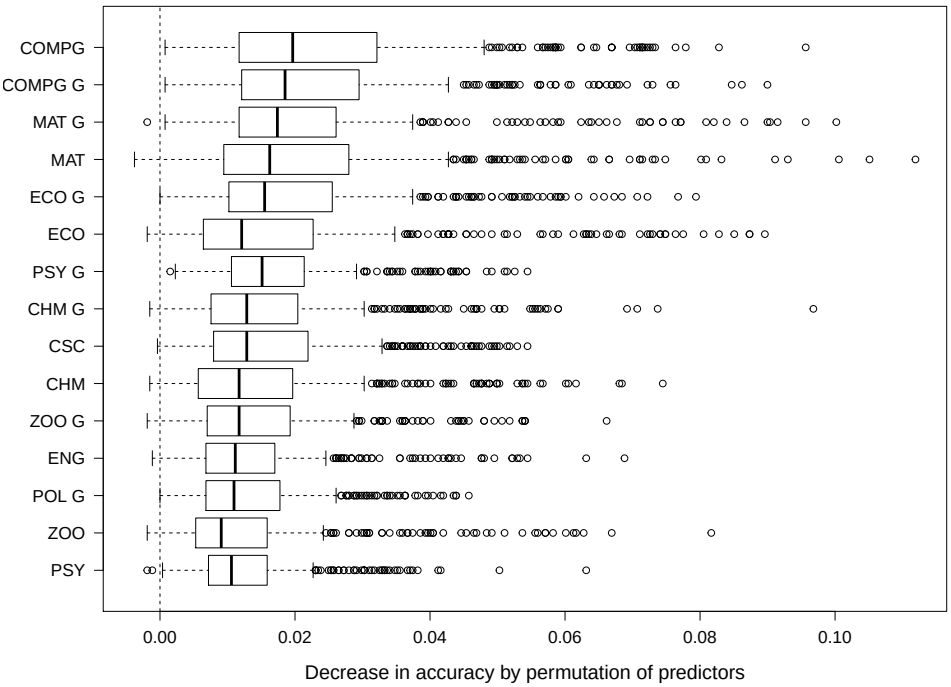
3.5 Conclusion

The first year's worth of courses and grades were used to build two classifiers; one that predicts if a student will complete their undergraduate program, the other that predicts the major of a student who completed a program. RFs were used to build those classifiers. Decision trees and RFs are the simple staple ML models and thus we wanted to gain experience with them before going into more modern machine learning algorithm. It felt fair comparing them to the statistical equivalent; linear models. RFs are easy to use with most statistical computing languages, fast to train, include high degree of interactions and they outperform linear logistic models in terms of prediction accuracy in this particular instance. For practitioners, RFs could be an alternative to typical linear models for various prediction tasks; to predict the number of students registered in second-year courses, the distribution of students across the many programs or to identify students at risk of failing or dropping out.

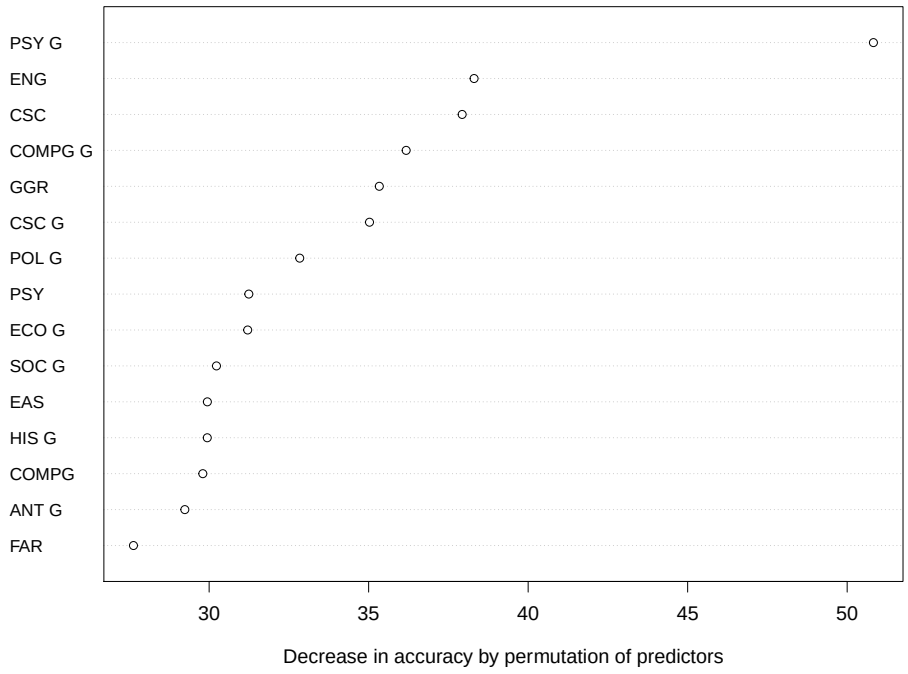
Evaluating the importance of each predictor is also something that offers RF in comparison to linear models. In this study, it was observed in Section 3.4 that grades were important for predicting if a student will complete their program. Grades in departments that were considered low-grading departments in some grades inflation research articles like Mathematics, Economics and Finance are consistently among the most important variables. These results indicate that a strong relationship exists between the grades in low-grading departments and the chance of succeeding at an undergraduate program, although this does not necessarily indicate a *causal* connection. Grades were somewhat less important predictors



(a) **Random forest # 1**.



(b) **Random forest # 2**.



(c) **Random forest # 3**.

Figure 3.2: Permutation decreases importance boxplots for the second research question.

for predicting the students' major but even though they were less important, grades in Mathematics, Finance, Economics and Psychology (PSY) were still frequently significantly important.

For potential improvements in the data analysis, we noted that some students might have completed more than one major or specialization. This might explain the relatively low accuracy for major choice prediction. Allowing for multiple major choices is a potential improvement for this model. This is in fact a multi-label classification problem and some solutions have already been proposed to adapt decision trees to accommodate this more complicated problem [33, 29, 30]. Some departments also share a great deal of similarities and might be considered equivalent by the university, thus combining some of them might increase the prediction accuracy.

The missing values in the predictors were also problematic. Ideally, the algorithm would consider splitting on the grade variables for a certain department only to classify students who took courses in that department. Developing a new decision tree algorithm where new variables are added to the pool of potential split variables depending on previous partitioning is the topic at hand in the next chapter of the thesis where we present BEST our implementation of such Decision Tree algorithm.

Chapter 4

BEST : A new decision tree algorithm that handles missing values

The main contribution of this chapter is the development of a new decision tree algorithm. The proposed approach allows users to guide the algorithm through the data partitioning process. This feature has many applications but in this chapter we demonstrate how to utilize this algorithm to analyse data sets containing missing values. We test our algorithm against various simulated data sets with different missing data structures. We also test our algorithm against a real data set, the data set introduction in the previous chapter.

In this chapter, the contributions are (1) a new decision tree algorithm publicly available on CRAN under the package named *BESTree*, (2) the demonstration that this algorithm efficiently handles missing values and produces results that are slightly more accurate and more interpretable than most common procedures without any imputations or pre-processing and (3) an updated analysis of the grade data set using this new algorithm. The main contributions of this chapter were introduced first in our article *BEST : A decision tree algorithm that handles missing values* [11] published in *Computational Statistics*.

4.1 Introduction

Machine learning algorithms are used in many exciting real data applications, in the chapter we address a classic statistics problem that is common when dealing with real data: predictors with missing values. Imputation techniques are designed to handle data with missing value under the assumption that data is missing completely at random (MCAR). Since this is a restrictive assumption we propose a solution to this problem that uses the tree structure of Classification and Regression Trees (CART) [22] to deal in an intuitive manner with observations that are missing in patterns which are not completely at random.

Our proposed new tree construction procedure was inspired by the data set introduce in Chapter 3 where the missing pattern of one subset of predictors could be perfectly explained by another subset; the *number of credits* variables explain the missing pattern of their respective *average grade* variables.

A typical decision tree is an algorithm that partitions the predictor space based upon a predictor value, splitting it into multiple subspaces and repeats this process recursively. Our proposed algorithm allows the researcher to impose a structure on the variables available for the partitioning process. By doing so, we construct Branch-Exclusive Splits Trees (BEST).

When a predictor X_j contains missing values, we can use other predictors to identify the region where the predictor X_j contains no missing value. This way, we can use the proposed algorithm to consider splitting on a predictor only when it contains no missing value based on previous partitioning. BEST can be easily adapted to any goodness of split criterion, stopping rule, labelling rule and any forest forming procedure [18, 20, 53]. BEST also has other applications; it can be used by researchers that would like to utilize some knowledge they have on the data generating distribution in order to guide the algorithm in selecting a more accurate and more interpretable classifier.

In this chapter, we introduce the proposed algorithm and some motivating examples before explaining in detail how the algorithm functions. We then do a quick review of the literature to position our algorithm within the current literature. Finally, some tests are performed on simulated data sets and on the real data that inspired this new algorithm.

4.2 Missing values

Let us now introduce the definition of missing data we are using in this chapter. As described in Section 2.1, a standard assumption in data analysis is that all observations are distributed according to the true data generation distribution $\mathcal{D}$. We could think of the missingness itself as a binary random variable $\mathbf{M}$ also of dimension m that is distributed according to some missingness generating distribution which is a part of $\mathcal{D}$, i.e. $\mathbf{X} \times \mathbf{M} \times \mathbf{Y} \sim \mathcal{D}$. If $\mathbf{M}$ represents the missingness of the vector of predictors $\mathbf{X}$ it means that $M_j = 1$ if X_j is observed and $M_j = 0$ if X_j is missing.

Three different relationships between $\mathbf{M}$ and $\mathbf{X}$ were defined by Rubin [126] and by Little and Rubin [98]. Seaman [132] later untangled the many definition inconsistencies of these relationships. In this chapter, we rely on simple definitions for an easy understanding of the structure we consider. First, missing completely at random (MCAR) is the simplest structure we consider: $\mathbf{M} \perp \mathbf{X}$. In other words, we consider the data is MCAR if the set of missing patterns M is independent of the set of predictors.

Missing at random (MAR) is much more complicated; it means that the missingness $\mathbf{M}$ is independent of missing observations but can still depend on observed predictors. More specifically, we define $\mathrm{X}^o = \{x_{ij} \in S | x_{ij} \text{ is observed}\}$ as the set of all observed predictors value, and $\mathrm{X}^{na} = \{x_{ij} \in S | x_{ij} \text{ is missing}\}$ as the set of missing predictors value. We say that data is MAR if the distribution of the missingness is conditionally independent of missing values given observed values : $\mathbf{M} \perp \mathrm{X}^{na} | \mathrm{X}^o$. As pointed by Seaman [132], MAR has not always been used consistently and the definition above is the one we settled on for this project. Note that MCAR implies MAR.

Finally, if the missing data is neither MCAR nor MAR, we say that the data is missing not at random (MNAR). In the sections to come we compare algorithms under these three missing data structures and we see that the relationship between $\mathbf{M}$ and $\mathbf{X}$ has a considerable effect on the performance of the missing values techniques that exist.

4.3 Branch-Exclusive Splits Trees (BEST)

We now introduce the proposed algorithm, BEST. The purpose of BEST is to utilize the tree structure itself in order to manage some missing data or some special structure among predictors.

4.3.1 Motivating Example

The data set previously introduced in Chapter 3 motivated our proposed algorithm and thus we reintroduce it quickly in the section. Recall that it contains information regarding the academic performances of students. The data set was provided to us by the Univeristy of Toronto. The predictors represent the number of credits and grades obtained in all the departments during the first two semesters. Table 4.1 contains a preview of the data with a reduced number of departments.

Student ID	Credits Math	Grade Math	Credits Econ	Grade Econ	Credits Hist	Grade Hist
101	2	72	3	88	0	NA
208	0	NA	0	NA	5	78

Table 4.1: An example of the motivating data set for 2 students and 3 departments

A student has no grade in many departments as they can only register to a limited number of courses within a year. In this situation, many grade variables are missing for every student. BEST handles that problem by considering the averaged grade obtained in a department only for students who took courses in that department.

For example, BEST forces the classification tree algorithm to split upon the *Number of credits* predictor to begin. Then, suppose *Number of credits in Statistics* is selected and 2 is the split point for the partitioning, BEST then allows splits on the *Grade in Statistics* predictor for the group of students in the region defined by *Number of credits in Statistics* > 2. Therefore, the *Number of credits* variables are used to define the region where their respective *Grade* variables are available for the partitioning process and thus we say that the *Number of credits* are gating variables for the *Grade* variables. The many *Grade* variables are gated by their respective *Number of credits* variables. Figure 4.9 illustrates how a BEST decision tree uses *Number of credits* as a gating variable for *Grade*.

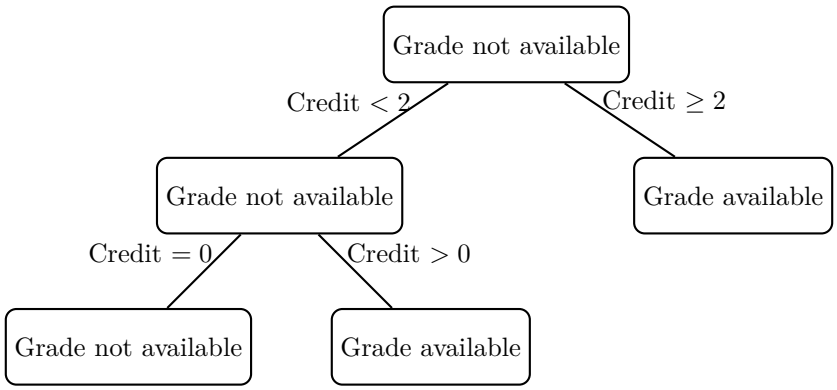


Figure 4.1: An example of BEST decision tree partitioning upon the Credit variable and the availability of the Grade variable within each of the produced regions

Other real data sets with similar problems are surveys. Many surveys have questions that are only relevant based on previous answers. Suppose question #1 is a yes/no question and is followed by : *If you answered no, please go to question #3.* This is quite typical and in that situation, BEST can use question #1 as a gating variable for question #2.

4.3.2 Intuition

As we explained in Section 2.1.2, a classification tree aims at partitioning the predictor space and labelling the resulting regions. CART does so by looking through all the possible splits and selecting the one that minimizes some pre-specified error measure. When using BEST some predictors are available to split upon only within some regions of the predictor space, such as the *Grade* variables in the motivating data set. These regions are defined according to other predictors, such as the *Number of credits* variables in the motivating data set. More generally, predictor X_l could be only available for the partitioning process in the region defined by $X_j < t$. We say that X_j is a gating variable for X_l or that X_l is gated by X_j. The variable X_l is not be available for the partitioning process until the gating variable allows it. Table 4.2 illustrates when BEST can partition the data using X_l based on a previous partitioning where BEST selects X_j as the splitting variable and s as the splitting value.

	$s < t$		$s > t$	
	Region $X_j < s$	Region $X_j \geq s$	Region $X_j < s$	Region $X_j \geq s$
X_l is available	Yes	No	No	No

Table 4.2: Availability of X_l if X_j is previously selected as splitting variable and s as splitting value

Doing so, predictors with missing values can be handled easily as BEST will partition the data according to that predictor only in regions where it does not contain missing value. If a data set contains missing values on predictor X_j but no predictor can help define the region with no missing value, we can add a new predictor X_{m+1} to the model as our gating variable. This new predictor is a dummy variable such that $X_{i,m+1} = 0$ if $X_{i,j}$ is missing and 1 if not. Doing so, we effectively add M_j as defined in Section 4.2, as a predictor in the model and thus will be defined as follows in the rest of the text. Then, BEST will only consider splitting on X_j in the subspace defined by $M_j = 1$ as illustrated in Figure 4.2. Multiple dummy variables are added to the model if multiple predictors contain missing values. Doing so also allows us to analyse the importance of the missing pattern M.

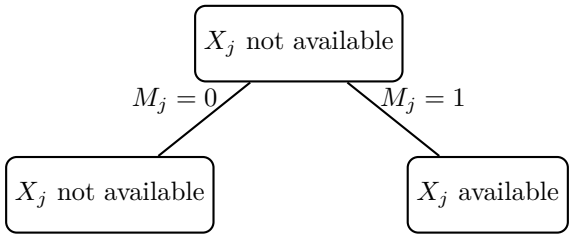


Figure 4.2: An example of BEST decision tree partitioning upon the dummy variable M_j that defines the availability of X_j

Finally, some insight on the data structure can be used to force some variable to be partitioned upon before others which is another application of BEST not described in this chapter. The result is a tree-structured classification model where some split variables are branch-exclusives. Even though we do not further mention it, the construction described below could be used for regression trees as well.

4.3.3 Algorithm implementation

Let us now discuss the current implementation of BEST. BEST takes as input the full data set S, the tuning parameter β and a list containing the predictor availability structure V. First S is set as the root node, the first set of observations to go through the following steps. The algorithm verifies a set of conditions before proceeding with the partitioning process. The first condition (C1) is that the region contains more than β observations, this is the main stopping rule. Then, the next condition (C2) is that the observations in the region have different labels; this condition makes sure that the algorithm has a reason to partition the data. Finally, the last condition (C3) is that at least one of the available predictors takes different values among the observations in the region; this is to guarantee that the algorithm can actually partition the data.

If at least one condition is false, then the region is defined as a leaf (terminal node), a label is assigned to that leaf for prediction purposes and the partitioning process is stopped. Usually the class that represents the majority in a leaf is selected as the label for that region, but one could define different label assignment rules.

If all conditions (C1, C2 and C3) are respected then the partitioning process begins. The algorithm will go through all available predictors. For a predictor j, the algorithm will go through all possible partitions s of the region with respect to the predictor j and will compute the total impurity of the resulting two regions $n_{r_1}Q_{r_1} + n_{r_2}Q_{r_2}$. Any region impurity measure Q can be used. BEST then selects the predictor j and the split s that minimize the total impurity and creates two children regions by splitting the data according to s.

The last step is to update the list of available predictors for the children regions. There exist multiple possible structures that can contain this information but within the R-language [118] we have settled on a list V since lists are very flexible. However, any array-like structure could work and to further optimize the speed of our package we could consider alternatives in the future. To begin $V[1]$ represents the set of predictors available for the partitioning process in the root node. More specifically $V[1]$ is a vector of size m where $V[1][j] = 1$ if the jth predictor is available to be split upon in the root node and $V[1][j] = 0$ otherwise. For instance, for the data set introduced in Section 4.3.1, the vector $V[1]$ equals 0 for all the *Grade* variables since they are not available for the partitioning process initially.

In the meanwhile, further elements of the list such as $V[j+1]$ are defined for $j \in 1, ...m$ and they contain the necessary information to update the predictors available for further partitioning. If j is a gating variable, then $V[j+1]$ should reflect that and contain the information needed to update the variable available following a partitioning based on j. For instance, if j is a continuous predictor, $V[j+1]$ contains a threshold value and a list of variables that becomes available given the appropriate partitioning. For instance, in the example introduced in Section 4.3.1, each *Number of credits* is associated with the threshold value 0. When a partition on a *Number of credits* variable happens, the partition containing

the observations where the *Number of credits* is strictly greater than 0 gain access to the corresponding *Grade* variable as illustrated in Figure 4.9.

Here is a pseudo-code of the proposed algorithm :

Algorithm 5 : BESTree(S,β,V)

INPUT: Training set S, hyper-parameter β and availability list V.
0. Start with the entire data set S and the set of available predictors $V[1]$.
1. Check conditions C1, C2 and C3.
2. **if** any condition is false :
    Assign a label to the node and exit.
else :
    for j in all available predictors:
        for s in all possible splits:
            Compute total impurity measure.
    Select the variable j and the split s with minimum impurity measure.
    Split the node into two child nodes.
    Update the available predictors for both children nodes using $V[j+1]$.
    Repeat steps 1 & 2 on the two children nodes.
OUTPUT: Fitted BESTree

The resulting tree can be pruned and as mentioned earlier can be constructed with any splitting rule, any stopping rule and any label assignment rule. Since one of the goals of this new algorithm is to produce accurate but also interpretable models we did not discuss forests so far, but the proposed tree construction procedure can be used to build any type of forest as well. Our implementation is publicly available on CRAN under the package named *BESTree* or on the first author's website. Anyone can download and install the package, read the vignette and use our proposed algorithm for their own research.

4.3.4 Theoretical justification

Recall from Section 2.1, the loss of a classifier h is defined as

$$L_{\mathcal{D}}(h) = \mathbf{P}_{\mathcal{D}}[h(x) \neq y], \tag{4.1}$$

which is the probability under the true data generating distribution $\mathcal{D}$ that the classifier h misclassifies x. Since the data generating distribution $\mathcal{D}$ is unknown, then the empirical loss computed with the data set S is typically used as an estimator of the true loss

$$L_S(h) = \frac{|\{i \in [n] : h(x_i) \neq y_i\}|}{n}, \tag{4.2}$$

which is the proportion of misclassified observations in the training set S. Usually a set of classifiers $\mathcal{H}$ is selected in advance and most learning algorithms are trying to identify the classifier $h \in \mathcal{H}$ that minimizes the empirical loss $L_S(h)$. The set $\mathcal{H}$ was named the hypothesis class by Shai & Shai [133]. The true

loss can be decomposed in a manner to observe a bias-complexity tradeoff. Suppose $h_S = \underset{h\in\mathcal{H}}{\text{argmin}} L_S(h)$, then

$$\begin{aligned} L_D(h_S) &= \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + (L_D(h_S) - \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)). \\ &= e_{\text{app}}(\mathcal{H}) + e_{\text{est}}(h_S). \end{aligned} \tag{4.3}$$

The approximation error, $\min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) = e_{\text{app}}$, is the minimum achievable loss within the hypothesis class. The second term , $(L_D(h_S) - \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)) = e_{\text{est}}$, is the estimation error and is caused by the use of the empirical loss instead of the true loss when selecting the best classifier h. Since the goal is to minimize the total loss a natural tradeoff emerges from equation 4.3. A vast, large and complex hypothesis class $\mathcal{H}$ leads to a wider choice of functions and therefore reduces e_{app}, but the classifier is more prone to overfitting, which increases e_{est}. Inversely, a small hypothesis class $\mathcal{H}$ reduces e_{est} but increases e_{app}.

Our proposed algorithm aims at obtaining a better classifier by restricting the hypothesis class to a smaller one without increasing the approximation error. Though there is a bit of wishful thinking, we hope to effectively reduce the estimation error more than we increase the approximation error.

Suppose $\mathcal{H}_T$ is defined as the set of all tree-structured classifiers. Then, BEST is a new algorithm that aims to find the best classifier in a new hypothesis class $\mathcal{H}_B$ that contains all the tree-structured classifiers that respect a set of conditions regarding the order that variables can be partitioned upon. Therefore, we have $\mathcal{H}_B \subset \mathcal{H}_T$ (R1). Because the complexity of $\mathcal{H}_B$ is smaller than the complexity of $\mathcal{H}_T$ the estimation error of BEST is smaller.

Next, let us take a look at the approximation error : $\min_{h\in\mathcal{H}_B} L_{\mathcal{D}}(h)$. When using BEST, we make multiple assumptions on how the partitioning should be processed. For example, we assume it is better to partition the data using the missing indicator M_j before partitioning the data using X_j. Rigorously speaking, BEST rest on the assumption that the best tree-structured classifier among all classification tree $\mathcal{H}_T$ is contained within the set of tree-structured classifiers that respect the partition ordering defined when using BEST: $\mathcal{H}_B$. In other words, we assume $\underset{h\in\mathcal{H}_T}{\text{argmin}} L_D(h) \in \mathcal{H}_B$ (R2). Suppose S is a data set, $h_S(\mathcal{H}_T)$ is the classifier that minimizes the empirical loss on $\mathcal{H}_T$ and $h_S(\mathcal{H}_B)$ is the classifier that minimizes the empirical loss on $\mathcal{H}_B$, then

$$\begin{aligned} L_D(h_S(\mathcal{H}_T)) &= \min_{h\in\mathcal{H}_T} L_{\mathcal{D}}(h) + e_{\text{est}}(h_S(\mathcal{H}_T)). \\ &= \min_{h\in\mathcal{H}_B} L_{\mathcal{D}}(h) + e_{\text{est}}(h_S(\mathcal{H}_T)). \qquad \text{by (R2)} \\ &\geq \min_{h\in\mathcal{H}_B} L_{\mathcal{D}}(h) + e_{\text{est}}(h_S(\mathcal{H}_B)). \qquad \text{by (R1)} \\ &= L_D(h_S(\mathcal{H}_B)), \end{aligned} \tag{4.4}$$

which implies that the under the assumption we have made, not only we would manage missing values but also reduce the loss. If our assumption $\underset{h\in\mathcal{H}_T}{\text{argmin}} L_D(h) \in \mathcal{H}_B$ is false, then we might increase the loss. Since the assumption itself is impossible to verify, the behaviour of the algorithm under multiple scenarios will be tested in Section 4.5 with simulated data.

4.4 Related work

Decision trees are well-established and a wide variety of solutions has already been proposed to handle missing values. In this section, we position our contribution within the current literature. We briefly introduce the current missing value techniques that are paired with decision trees and we establish the main differences between these techniques and the proposed algorithm introduced in Section 4.3. To do so, we use recent surveys [129, 146, 39, 51] that defined and compared these techniques using various simulated and real data sets.

Predictive value imputation (PVI) methods are popular approaches to deal with missing values. They estimate and impute the missing values within both the training and the test set. The simplest imputation consists of replacing the missing values with the mean for numerical predictors or the mode for categorical predictors. More advanced prediction models have also been proposed, such as linear model, k-nearest neighbours or expectation-maximization (EM).

Since these methods use known predictors to impute values for the missing ones, if the predictors are uncorrelated these approaches have no predictive power. This leads to poor imputations and it is a major drawback noted by Saar-Tsechansky and Provost [129] and Gavankar [51]. Nonetheless, Twala [146] demonstrated using simulated data sets the great performances of expectation-maximization multiple imputations (EMMI) [131]. This imputation technique produces multiple different imputations based on expectation-maximization and then aggregates the results.

Our proposed algorithm differs from imputation methods as it only uses known information to build the classifier instead of using potentially unreliable prediction to replace missing values.

The surrogate variable (SV) approach [22] is a special case of predictive value imputation. As explained by Hastie et al. [61], during the training process, when considering a predictor for a split, only the observations for which that predictor is not missing are used. After the primary predictor and split point have been selected, a list of surrogate predictors and split points is constructed. The first surrogate split is the predictor and split point pair that best mimic the split of the training data achieved by the primary split. Then the second surrogate split is determined among the leftovers predictors and so on. When splitting the training set during the tree-building procedure or when sending and observation down the tree during prediction, the surrogate splits are used in order if the primary splitting predictor value is missing.

Many articles [45, 129, 146, 39] showed that the results are not satisfactory in many cases and Kim and Loh [82] noted the variable selection biased caused by this approach. Our proposed approach is much more computationally efficient and utilizes the missing pattern as a predictor instead of ignoring it.

Reduced-feature models are suggested by Saar [129] when missing values appear only in the prediction process. When we need to classify a new observation, a tree is built using only the known predictors of the new observation. If multiple observations contain different missing pattern then multiple trees are built to classify the various observations. It shares a great deal of similarities with lazy decision trees [50] as both models tailor a classifier to a specific observation and uses only known predictors to do so.

A major drawback of this technique is that it only manages missing values during prediction while our proposed technique can handle missing value for both training and prediction. BEST also differs from reduced-feature models as it not only uses the known values but also utilizes the fact that we know some predictors are missing instead of discarding this information.

The popular C4.5 implementation [117] has its own way to manage missing data, defined as a distribution-based imputation (DBI). When selecting the predictor to split upon, only the observations with known values are considered. After choosing the best predictor to split upon, observations with known values are split as usual. Observations with unknown values are distributed among the two child nodes proportionately to the split on observed values. Similarly, for prediction, a new test observation with missing value is split into branches according to the portions of training example falling into those branches. The prediction is then based upon a weighted predictions among possible leaves.

This technique is computationally slow and offers poor performances in terms of prediction accuracy according to some of the surveys we mentioned [129, 147]. Our technique should be faster, more accurate and more interpretable.

As described by Ding and Simonoff [39], the Separate Class (SC) method replaces the missing value with a new value or a new class for all missing observations. For categorical predictors we can simply define *missing value* as a category on its own and for continuous predictors any value out of the interval of observed value can be used.

This technique has the best performances according to Ding and Simonoff [39] when there are missing values in both the training and the test set and when observations are missing not at random (MNAR). Twala et al. [147] also came up with similar results with a generalization of the separate class method coined Missing Incorporated in Attribute (MIA). These techniques are by far the closest in spirit to BEST. As BEST, MIA and SC allow for similar data partitioning we do not expect BEST to offer a drastic improvement in terms of accuracy. On the other hand, BEST identifies the missing pattern using other predictors rather than including the missingness information within the predictor containing missing values. Doing so, our approach should offer more interpretable results and a more accurate variable importance analysis.

Finally, there exist many other articles discussing decision trees in the context of missing values. Some introduce ways to use decision trees and random forests as missing values imputation techniques such as Rahman and Islam [119] and others to identify the missingness structure such as Tierney et al. [142]. Our approach is different since we are not using decision trees to pre-process data with missing value or to identify the missingness structure but to perform a decision tree analysis of a data set that contains missing values.

4.5 Experiments : Simulated data sets

We now assess the abilities of our algorithm on simulated data sets. All of our experiments are done using the R-language [118]. In the following set of simulations, we will compare 6 methods; (1) BEST, our proposed approach, (2) the Distribution Based Imputation (DBI) proposed by Quinlan [117] implemented

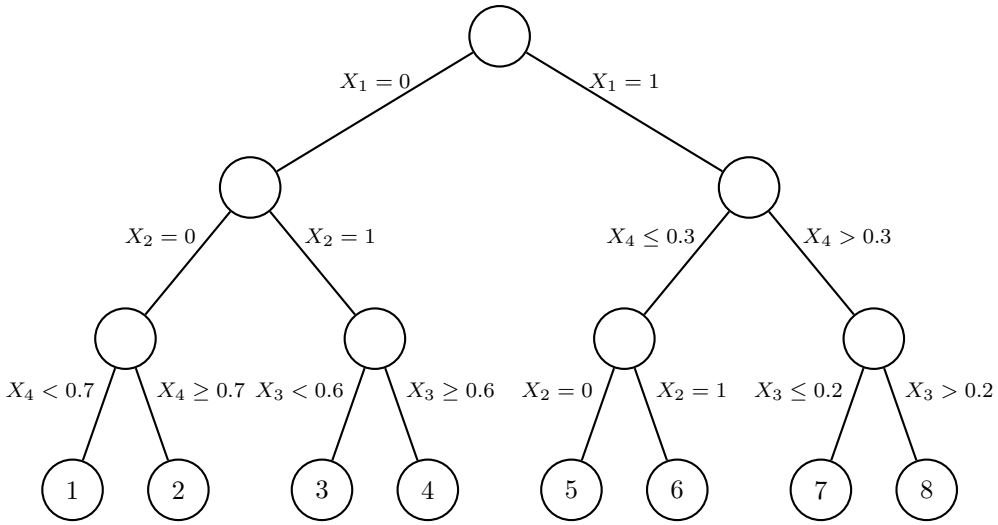


Figure 4.3: The decision tree used to generate our simulation data sets

in the *C5.0* package [90], (3) a simple single variable imputation (SVI), either the mode for a categorical predictor or the mean for a numerical one, (4) a refined predictive value imputation (PVI) using known predictors; EM for numerical predictors and multinomial logistic regression for categorical ones [149], (5) the separate class (SC) approach and finally (6) the surrogate variable technique introduced by Breiman et al. [22] implemented in the *rpart* package [141]. Since the Reduced-Feature Model was the least accurate in every single experiment we have done, we decided not to include it in the following plots to improve readability.

We will generate data sets containing 4 predictors; X_1 and X_2 are binary predictors and X_3 and X_4 are continuous predictors on $(0, 1)$. The response is categorical and can take up to 8 different values. The binary predictors are generated according to a Bernoulli distribution and the continuous predictors are generated according to a Uniform distribution. The response labels are assigned according to the tree in Figure 4.3 but a proportion of the responses labels are randomly assigned.

Let us describe the experimental procedure. We begin by generating a data set as we described in the previous paragraph and we apply a missing pattern to the data set; details are included in the respective subsections. Then we fit a pruned decision tree using each of the six methods mentioned earlier in the section. We experiments with various parameters such as the number of training observations, the variable containing missing values and the type of missing patterns. Finally, we compare the technique performances using the prediction accuracy on the test set where the accuracy is defined as the proportion of correctly classified observations.

4.5.1 MAR : Missingness depends on observed predictors

This first experimental set up is meant to test the missing pattern structure BEST is designed for. In this set up, the missing pattern of a predictor is fully explained by another, fully observed, predictor. The binary predictor X_1 was designated as gating variable for a randomly sampled predictor, either X_2,

X_3 or X_4. In our first experiment we randomly sample either 0 or 1 and the gated variable is missing if X_1 equals the randomly sampled value. This procedure is repeated 200 times for this experiment.

We present our results using Sina Plots [134, 152]. This allows to better visualize the distribution of the performances of the different techniques.

Figure 4.4a contains the results of our simulation using 200 training observations to build the classifiers. Figure 4.4b contains the results of our simulation obtained when using 1000 training observations.

Both of the plots convey similar information. When the missing pattern depends on other predictors, the performance of BEST is similar to many competitors. However, BEST leads to more interpretable decision trees and does not require any imputation. There is no notable differences between the results obtained with 200 training observations and 1000 training observations in that experiment. Going forward, we keep the number of training observations fixed to 200 but we vary other parameters.

In the next experiment we once again designate X_1 as the gating variable, sample one of the other predictors as the gated predictor and randomly sample either 0 or 1 to gate the randomly selected predictor. However, this time we generate a less extreme missing pattern. The gated variable will be missing with a probability of 50% when X_1 equals the randomly sampled value.

We can observe in Figure 4.4c that a less extreme missing pattern does not affect one technique more than others. Overall, BEST accuracy when the data is missing at random given other predictors is as good as other tested techniques. This is very positive considering BEST does nto require any special pre-processing which makes it easier to use than its counterpart.

4.5.2 MNAR : Missingness depends on missing values

Let us now proceed with an experiment where predictors are missing not at random. If a continuous predictor, let us say X_j, is randomly selected, then a random value t is drawn at random within the domain of X_j and serves as a threshold value. Finally, a Bernoulli variable b is drawn and if $b = 0$, then if $X_j < t$ X_j is set missing, otherwise if $b = 1$ then X_j is missing if $X_j \geq t$.

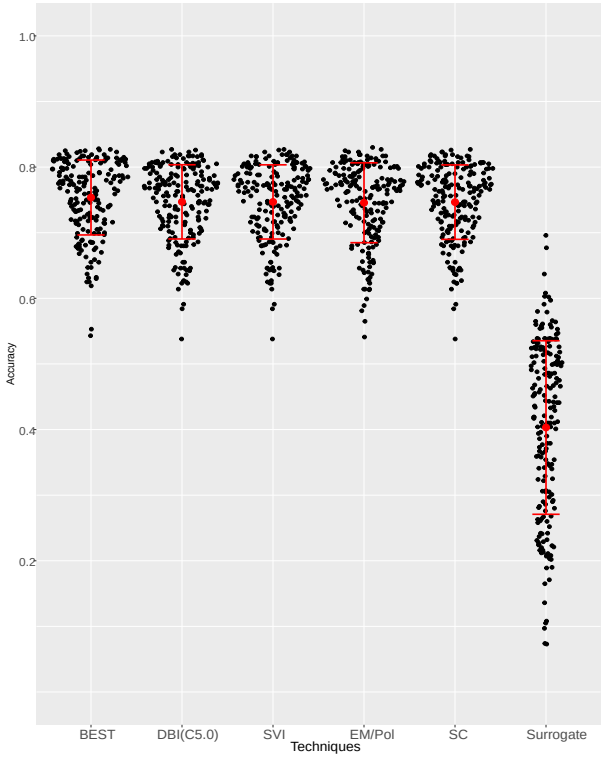
If one of the binary predictors is selected, then a Bernoulli variable is drawn and X_j is missing if X_j equals the Bernoulli variable. Since the missingness of X_j depends on the value of X_j itself, this is considered MNAR. This procedure is repeated 100 times.

In Figure 4.4d we observe that BEST outperforms DBI and multiple imputations. The performances of BEST are comparable to those of SVI but are slightly lower those of the SC approach.

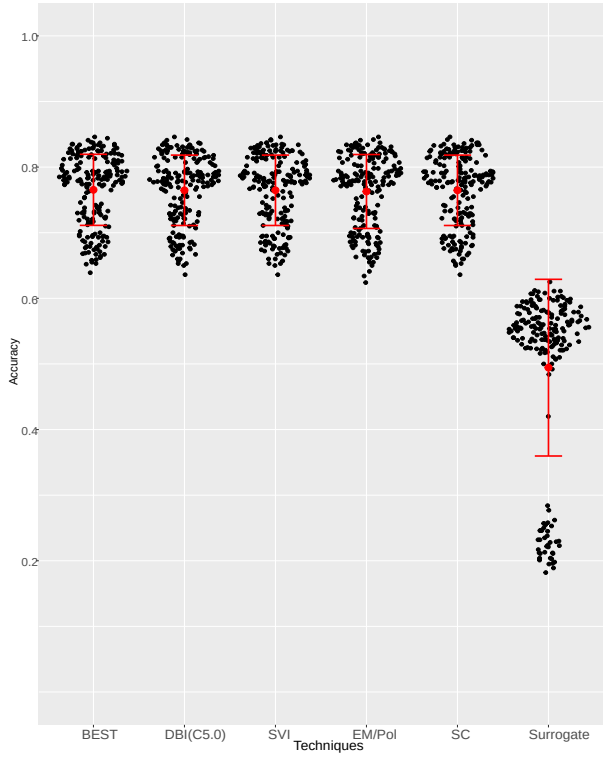
4.5.3 MAR : Missingness depends on the response

According to Ding et al. [39], the relationship between the missing pattern and the response variable has a great effect on the results obtained from different missing value treatments.

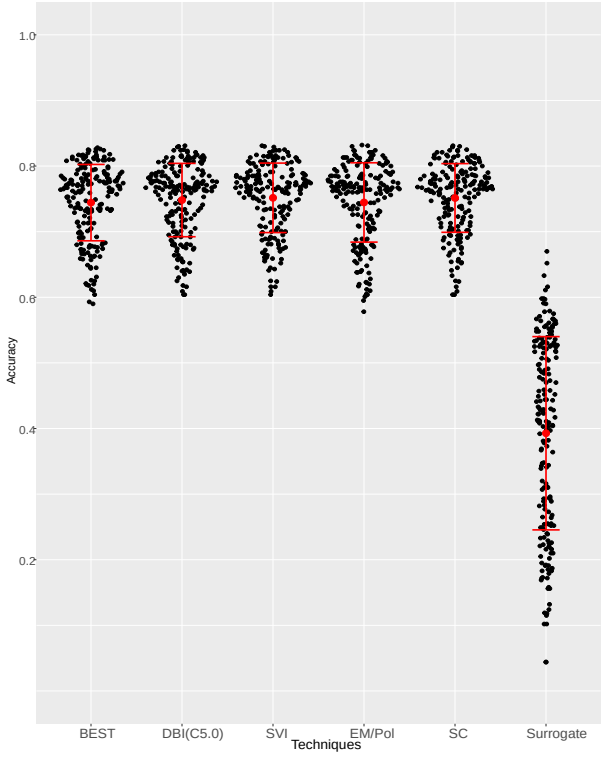
In this simulation, one of the predictors is randomly selected, let us say X_j, every iteration and the censoring process is then applied. The censoring process goes as follows; one of the eight response labels



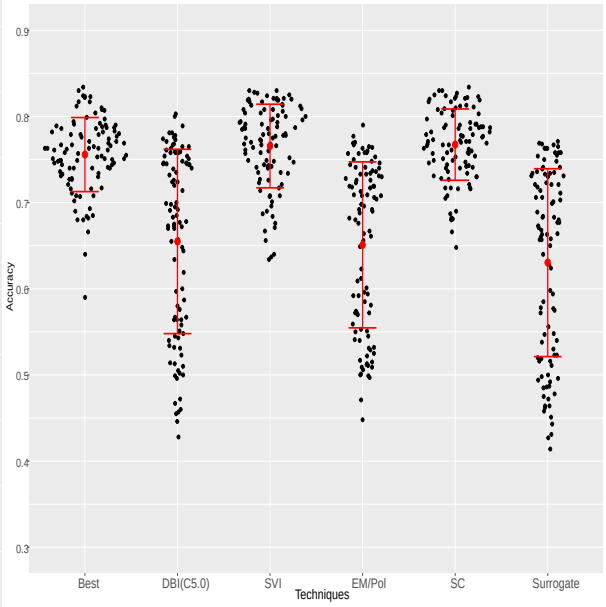
(a) Missing pattern depends on other predictors (200 training observations)



(b) Missing pattern depends on other predictors (1000 training observations)



(c) Half of the gated variable missing



(d) Missing pattern depends on missing values

Figure 4.4: Prediction accuracy for different techniques under different experimental set up.

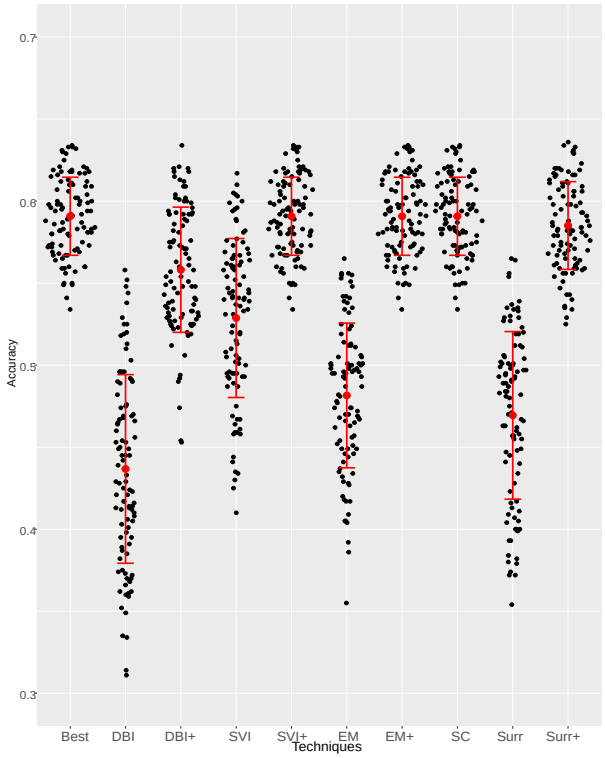


Figure 4.5: Accuracy by techniques when the missing pattern depends on the response.

is randomly selected, and X_j is missing for all observations with that selected label. In this experiment we have used a dummy variable as the gating variable for the BEST algorithm. This procedure will be repeated 100 times for this experiment.

In this experiment, the missing pattern is actually a variable with predictive power and therefore, models like BEST and SC shine as they utilize the fact that there is missing values instead of trying to impute them. BEST and SC approaches have similar results and their performances is higher than any other techniques as seen in Figure 4.5.

First, we noticed the high performances of the simple single value imputation in some cases; it performs well if the predictor with missing value is continuous. Our experiments reveals that when the predictor containing missing value is continuous, replacing missing values with the mean is equivalent to creating a separate class because only the missing values will exactly take the value of the mean. If the predictor with missing value is categorical, replacing missing values with the mode will make the observation with missing value undistinguishable from observations that truly belong to that class which leads to the poor results.

We also wanted to understand the impact of the indicator gating variable on the success of BEST. To do so, we added the indicator variable to the data set and we applied the various missing value techniques as well on top. Those are identified with the plus (+) sign.

In Figure 4.5 we can see that the indicator variable is important if the missingness depends on the response variable. The plus sign represents the result obtained by the technique with a data set that

Data	X_1	$\mathbf{X_2}$	X_3	X_4	$\mathbf{M_2}$
Complete	112.37705	**26.76542**	158.55069	102.91328	-
BEST	94.53228	**10.80073**	155.39201	78.14924	**156.27593**
SC	103.62706	**171.70715**	147.89474	80.51999	-
SC+.	94.36789	**81.83499**	128.78712	55.14764	**84.45193**

Table 4.3: Variable Importance table computed using the GINI decrease importance

also includes the indicator variable. For example, for SVI+, we both imputed the missing values using SVI and we included the indicator variable in the data set. Adding this variable to the data set lead to improved performances for all of the tested techniques. BEST offers great results considering it does not need to impute the missing values nor any other data pre-processing. If the missing indicator variable is to be added to the data set, it is counter-intuitive to also impute missing values. We argue in Section 4.6.3 that BEST also produces trees that are more interpretable and that BEST leads to a more reliable variable importance analysis than other algorithms. For those reasons, we believe BEST is the best approach when the pattern is missing at random given the response.

4.5.4 Random forests and variable importance

Let us now build a small example where random forests are used to analyse the variable importance. Random forests are popular in exploratory analysis [139] as the variable importance tools developed for this model became quite popular.

As seen in the previous experiments, when BEST performs well, so does the SC approach usually since both of these techniques lead similar trees. In this section we quickly discuss how BEST produces more accurate variable importance computations than the SC approach. We created an example where the missing pattern depends on the response, used either the SC approach or BEST to handle missing values and built a forest with those trees.

When the values for a predictor are conditionally missing at random given the response, the missing pattern is itself a good predictor. We believe it is important that a variable importance analysis distinguishes between the importance of the predictor with missing value, say X_j, from the importance of its missing pattern M_j. A random forest of trees built under the SC approach fails to distinguish between the effect of the observed values of the predictor from the effect of the missing pattern. Since BEST actually uses a variable to define the region with missing values, either with another predictor or a user-created dummy variable, then this gating variable importance will better represent the predictive power of the missing pattern.

We built a random forest using the complete data set and computed the GINI decrease importance. Then we have randomly selected one of the eight labels, and the predictor X_2, a predictor of low importance according to the GINI decreases under the complete data set, is rendered missing depending on the value of the response. Since the SC approach uses the predictor containing missing value to identify observations containing missing value then it identifies X_2 as the most important predictor. Since we built this problem we know X_2 is an unimportant predictor, this is also support by the complete data variable importance. It is actually the missing pattern M_2 that contribute to the model performance. We

included the missing pattern M_2 to the data set and used the SC approach again (SC+). Unfortunately, the SC approach still grossly overestimate the true importance of the predictor X_2.

Using BEST, we can easily observe that the missing pattern, M_2 is the important predictor and that X_2 is actually of low importance when observed as it should be according to the complete data variable importance. We believe the variable importance analysis produced by BEST better reflects the true predictive power of the various predictors and this is a great benefit from using BEST over the SC approach.

4.5.5 Simulations: takeaways and limitations

Throughout various simulation experiments we have been able to highlight the success of BEST under various scenarios. Since the SC approach does not impute the missing values then it can create partitioning similar from those created by BEST. However, the SC approach does not need to first isolate the missing value in order to partition upon the variable containing missing values which can sometimes be valuable. On the other hand, we have argued that BEST produces a variably importance that is more accurate than its SC counterpart. We also elaborate in a later section that BEST produces trees that are more interpretable.

Our simulation also revealed that BEST suffers from a weakness when the gating variable is of low importance. This can happen if only a small proportion of data is missing, if the missing pattern is simply non-informative or in some cases when data is MCAR. In those cases, BEST will rarely partition upon the gating variable and thus will struggle to partition upon the gated variables which will almost surely reduce the accuracy of the resulting tree. This weakness is intrinsic to the algorithm as it is caused by the greedy nature of decision trees which are usually fitted by growing large trees and pruning them later. During the partitioning process, a classic decision tree approach only sees the reduction in impurity gained with a single partition and thus cannot perceive the accuracy gained by the combination of two successive partitioning.

Building a random forest of BEST trees instead of using a single tree could alleviate that problem. When we build trees in a forest, predictors are randomly selected and thus the algorithm will partition on the gating variable of low important from time to time which will reveal the important gated variable. Another way to circumvent this limitation would be to consider pairs of consecutive splits but this would come at a great cost; this would drastically increase the run time of the algorithm.

Another perspective on this issue is with respect to the splitting criteria. As of now, the splitting criteria is one of a traditional decision trees, the criteria is meant to decrease the impurity of the response. Consequently, it is passive towards missing patterns; the criteria is the same for gating variable and gated variable. A suggestion we received from our external examiner is to consider a splitting criteria that favour splits that improve availability of new predictors. This is an idea we would like to pursue in a future project, the conclusion contains a few ideas on how to implement such criteria.

A limitation of our simulated experiments is the absence of the MIA algorithm [147] which shares a lot of similarities with BEST. No implementation or package of MIA was available at the time we ran this experiment and thus an advantage of BEST is that a R package is available for researcher looking

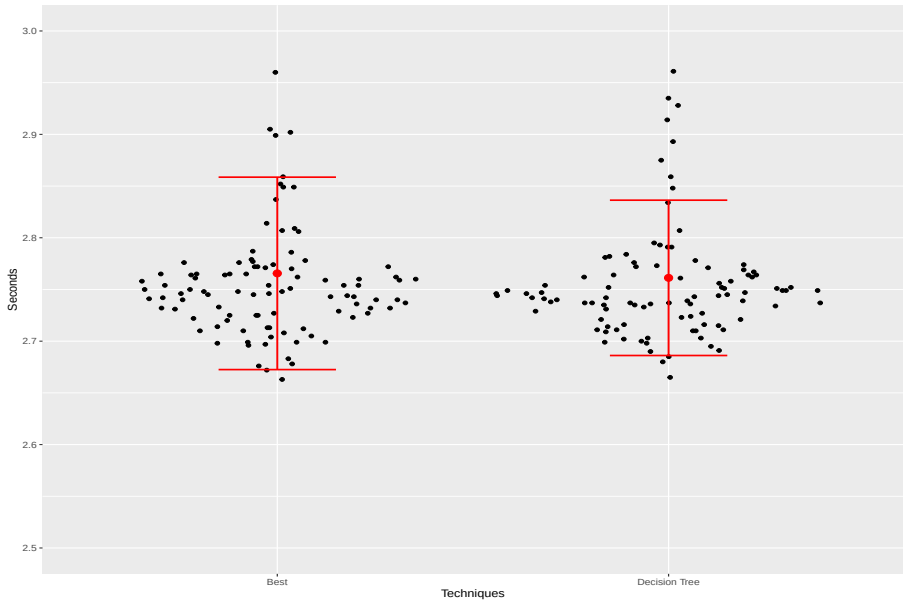


Figure 4.6: Run time of BEST compared to a decision tree.

for an algorithm ready to use off-the-shelf. Regarding prediction accuracy, we do not expect BEST to outperform MIA as they can produce similar partitioning but we expect BEST to be slightly faster as MIA greatly increases the number of operations when building the tree. Finally, as discuss earlier we do believe BEST produces more interpretable trees than the SC approach, the same argument holds when comparing BEST to MIA. The interpretability argument is expanded in Section 4.6.3.

Finally, another limitation of our experiment is that we did not test the run times of the algorithms in an extensive manner. We think there was no way to build a fair comparison among the various techniques. As it stands our package BESTree is entirely coded in R and thus currently runs more slowly than than popular decision tree packages such as the *C5.0* package [90], the *rpart* package [141] or the *party* package [65].

We believe that the speed difference is caused by our suboptimal coding and the language used but not by the structure of our proposed algorithm. Our BESTree package contains a regular decision tree function which shares most of its architecture with the BEST function but without the new functionalities introduced by BEST. We tested the run speed of our proposed algorithm BEST against the regular decision algorithm included in our package.

We generated 200 samples of 5000 observations and computed the run time in seconds needed to fit both BEST and a regular decision tree algorithm. As observed in Figure 4.6, our intuition was right; the added functionality of BEST does not increase the run time significantly. In order to improve the run time of all functions in our package, we plan on coding some key components in a faster language such as C++ in a future release.

# of obs	5000		10000		15000		20000	
Methods	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
DBI	0.7223	0.0035	0.7336	0.0038	0.7385	0.0092	0.7402	0.0095
SC	0.7307	0.0066	0.7387	0.0044	0.7427	0.0036	0.7462	0.0033
Surrogate	0.7291	0.0053	0.7301	0.0046	0.7311	0.0034	0.7300	0.0032
BEST	0.7333	0.0062	0.7424	0.0045	0.7457	0.0037	0.7479	0.0033

Table 4.4: Mean accuracy and standard deviation when predicting program completion

# of obs	5000		10000		15000		20000	
Methods	Mean	S.D.	Mean	S.D.	Mean	S.D.	Mean	S.D.
DBI	0.3581	0.0055	0.376	0.0043	0.3871	0.0039	0.3936	0.0057
SC	0.3992	0.0079	0.4172	0.0069	0.4262	0.0047	0.4332	0.0044
Surrogate	0.3639	0.0121	0.3661	0.011	0.3723	0.012	0.3742	0.012
BEST	0.3952	0.0074	0.4164	0.0051	0.4265	0.0043	0.4319	0.0043

Table 4.5: Mean accuracy and standard deviation when predicting the major

4.6 Experiments : grades data set

4.6.1 Predicting program completion

The data set mentioned in Section 4.3.1 was analysed using BEST. Once again, the accuracy of the proposed algorithm is compared to other techniques that handle missing values. To begin, we predict if a student completes its program using its first year of courses and results. Then we predict the major of those who completed a program. The data set contains 38842 observations. Our set of predictors consists of the number of credits attempted in all the departments and the average grade obtained in those respective departments. The number of credits is a numerical variable that serves as the gating variable for the respective grade. If the number of credits attempted in a department is greater than 0 for every observation in a region then BEST acquires access to the grade variable. We have randomly sampled training sets of different sizes and used all the remaining observations to assess the accuracy. This process was repeated 15 times and we have averaged the results. We did not include the single value imputation because we expect this technique to produce the same result as the SC approach since all predictors are numerical. We did not include the imputation produced by the mice package [149] as the package had issues with with our large data set. Finally, we used tables to show our results because with only 15 trials the Sina plots were not informative.

In table 4.4, we observe that BEST produces the most accurate decision trees for that data set for all training data sizes. This improvement is important as it is essential for universities to predict if a student is at risk of not completing its program. This information is valuable since most universities want to help their students move forward by adjusting their policies or providing them the resources needed. We have done most of our experiments with trees, but using a random forest of BEST trees the prediction accuracy reaches 79.89%, which is higher than anything previously obtained in chapter 3 with random forests made with traditional decision trees.

Another reason why we might prefer using BEST in this analysis is its ability to rightfully identify the variable importance. As we discussed in chapter 3 we were interested in the importance of the predictors. Therefore BEST is an improvement as it truly identifies the importance of the gating variables, the number of credits, as we have shown in Section 4.5.4. In this case we were able to distinguish the importance of the number of credits in a department from the importance of the grade obtained in that department which was impossible before the implementation of BEST. Figure 4.7 contains the variable importance plots for both research questions.

Figure 4.7a illustrates the results obtained when we evaluated the variable importance using the averaged GINI decrease based on a forest of 200 BEST trees. The results are quite different from what we previously obtained in chapter 3. It does seems like overall the number of credit is the most important variable though the important departments are mostly the same; ASSEM, MAT, ECO, ENG, CHM were previously deemed important and still are. The number of credits attempted in the first year seminar courses (ASSEM) is the most important predictor. In this case, taking a first-year seminar course was positively correlated with succeeding an undergraduate program. These seminars were brand new at the University at the time the data was collected and this analysis provides evidence of the merit of these courses to establish a student's profile.

4.6.2 Predicting the major

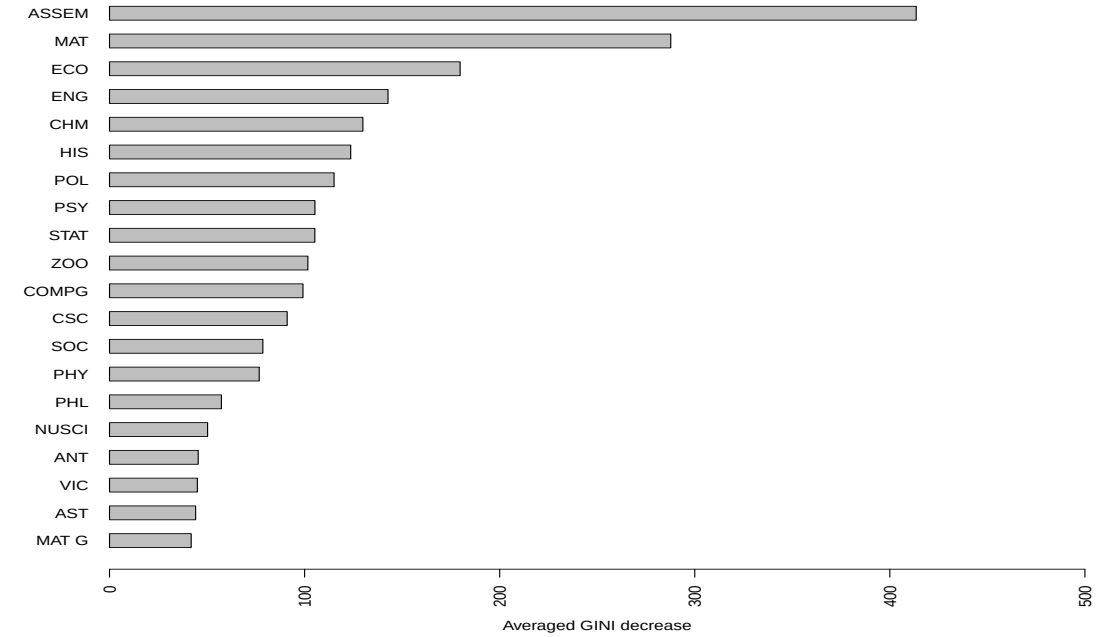
In the second analysis, we look at the 26488 students who completed their program. Using the same set of explanatory variables, we predict the department they majored in. Predicting the major that will be completed by students can help Universities plan ahead the resources needed by each departments. According to the results in table 4.5 BEST can accurately produce such predictions.

In table 4.5 we observe closer results from the two best-performing algorithms. BEST and SC have almost indistinguishable performances and are the top performers. A random forest of BEST trees reaches an accuracy of 47.57% which is slightly higher than previously obtained results in chapter 3. Even though the results are a lot closer between BEST and SC, our proposed algorithm produces trees that are more interpretable and can be used to produce a non-biased variable importance analysis as argued in Section 4.5.4.

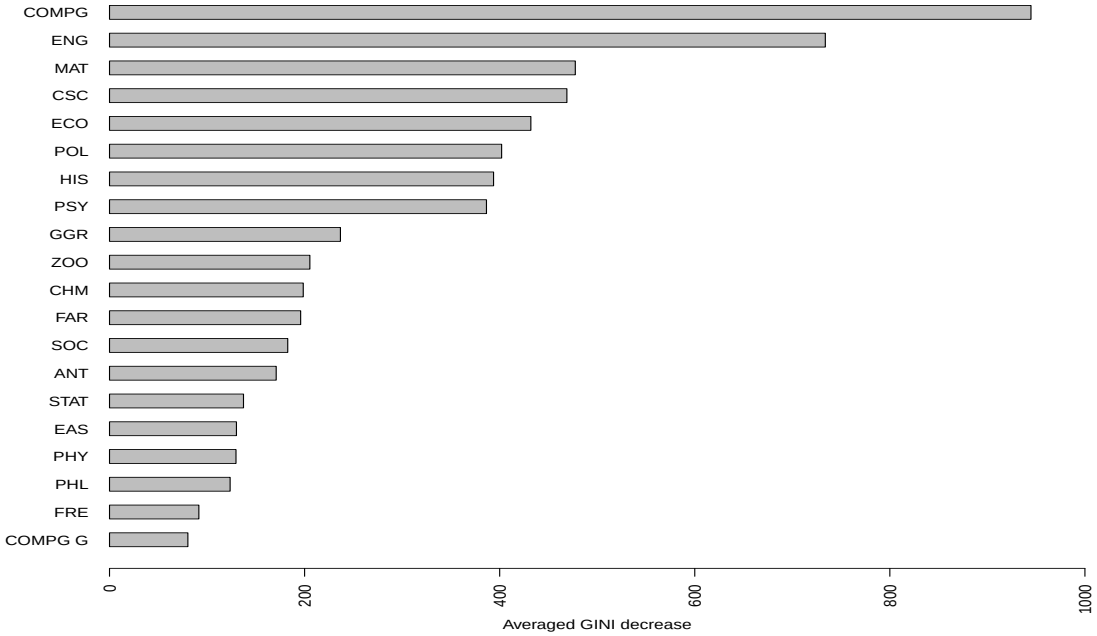
Figure 4.7b contains the variable importance we obtained. In this particular case the results were not significantly different from those previously obtained though once again we observe a slight drop in importance of the grades variables. The number of credits in the Finance department (COMPG) and the number of credits in the English department (ENG) have relatively high importance compared to all the other predictors. We observed that the students who obtained a major in either of those were very likely to register to many courses in these respective departments starting in the first year. The number of credits in the Mathematics department (MAT) and the number of credits in the Computer Science department (CSC) are also of noticeably high importance. We have noticed that these variables are quite useful to predict if a student picks a major in a scientific field.

4.6.3 Improved interpretability

We mention throughout this chapter that BEST leads to more interpretable decision trees than the SC approach. Let us now demonstrate this with the grade data set. It provides a good example.

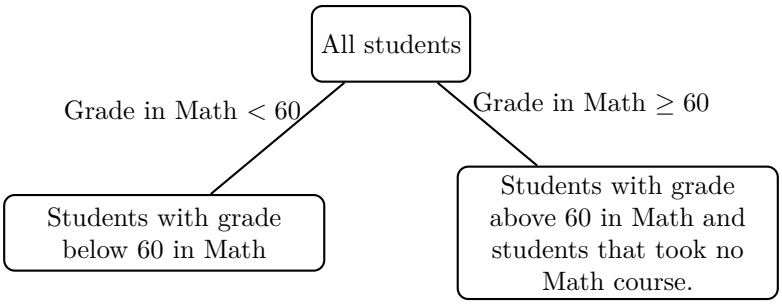


(a) Predicting program completion.

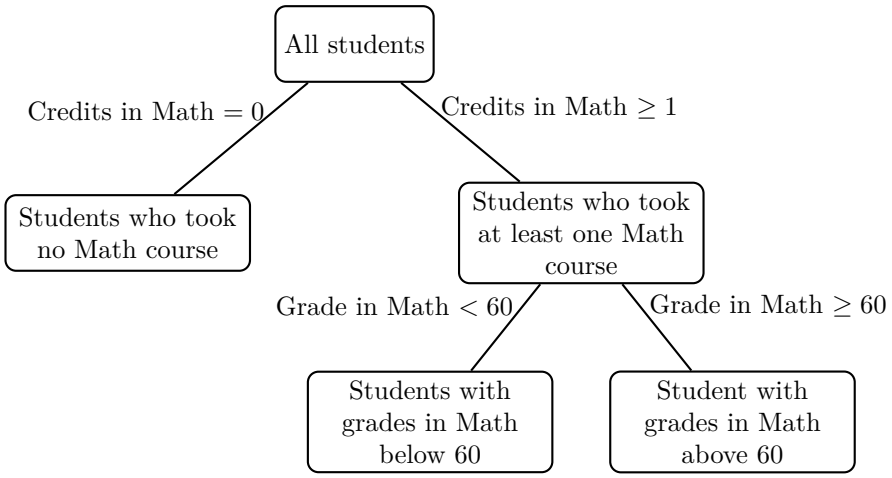


(b) Predicting the major completed.

Figure 4.7: Variable importance plots .



(a) An illustration of a decision tree partitioning produced by the SC approach and the associated regions.



(b) An illustration of a decision tree partitioning produced by BEST and the associated regions.

Figure 4.8: Illustration of BEST's improved interpretability.

For the SC approach we have replaced missing grades by a value outside of the domain, 101. Frequently in our experiments, the tree constructed under the SC approach partitions upon the grade in departments before the number of credits in the same department. For example, if *Grades in Mathematics* is the first split variable selected and 60 is selected as the split point, then the SC approach produces the partitioning illustrated in Figure 4.8a.

It is hard to extract interpretable information out of the tree in Figure 4.8a. Does this partition imply students with no experience in Mathematics behave similarly to students with good results in Mathematics ? BEST achieves higher accuracy while keeping the partitions logical and interpretable. BEST begins by partitioning students who attempted at least 1 credit in Mathematics from those who did not. Then, among students who attempted at least 1 credit in Mathematics, BEST will partition them according to their grades, which leads to a more interpretable sequence of partitions as illustrated in Figure 4.8b. More partitions are needed for similar data partitioning but the produces trees are much more interpretable. To sum up, BEST leads to more interpretable trees then its closest competitor in terms of classification performance, the SC approach.

That being said, BEST produces trees as large as any decision tree algorithms because the stopping rule is the same. Thus, in some cases they might be so large that it is impossible to look at the entire tree to interpret the classification mechanism. In the experiments of Sections 4.6.1 and 4.6.2 the trees

produced are rather big, containing on average 150 terminal nodes. In those cases it is possible to look at the first few splits to understand some global partitioning rules that were applied to large amount of data. In the experiments of the previous section, the majority of the fitted trees shared the first few levels. We included in the appendix the first 4 levels of a tree obtained when predicting program completion.

4.6.4 Real-world data set experiment takeaways

Even though we experimented on a single real-world data, the results are extremely positive. BEST has higher or similar performance than the other tested algorithms. BEST produces more accurate variable importance analysis and more interpretable trees than the SC approach, BEST's closest competitor. Finally, to use BEST we did not need to do any imputations beforehand which is another reason why we prefer BEST.

4.7 Conclusion

We have constructed a modified tree-building algorithm that lets the users decide the regions of the predictor space where variables are available for the data partitioning process. We explained how to use this feature to manage missing values. BEST has the elegant property of analysing a variable only when values are known without assuming any missingness structure. It produces highly interpretable trees and achieves comparable accuracy to most missing value handling techniques in cases we have identified using simulated data sets. Even though BEST shares similarities with the separate class technique, BEST leads to a more accurate variable importance analysis and produces more interpretable and intuitive trees.

BEST suffers from a weakness when the gating variable has no predictive power. In those cases, the algorithm never chooses to split upon the gating variable and thus is never allowed to use the branch-exclusive variable. In a future project, we would like to implement a new splitting criteria that favour splits on gating variables. One way to do so would be to weight the criteria of a split by the number of observations for which additional predictors would be available given that split. Doing so would not only favour splits on gating variable, generally speaking, but would also favour splits closer to the threshold value and favour splits on gating variables more strongly earlier than later during the partitioning process.

In the simulated experiments we have performed, results were mostly positive as BEST outperforms some other techniques when data is MAR and MNAR. The results produced by BEST were also impressive when the algorithm was used on the real motivating *grades data set*. We achieved higher accuracy than with most other techniques while obtaining a more interpretable classifier. Since variable importance is a concern in the *grades data set* analysis, BEST is an improvement as it answers that research question by providing a more reliable variable importance analysis than the separate class approach previously used in chapter 3.

Chapters 3 and 4 discussed how we were first introduced to machine learning, with well-established algorithms such as decision trees and random forests. We explored these models from an applied and a

theoretical perspective by completing a data analysis and releasing an new algorithm of our own. In the following chapters we explore more recent models.

Appendix

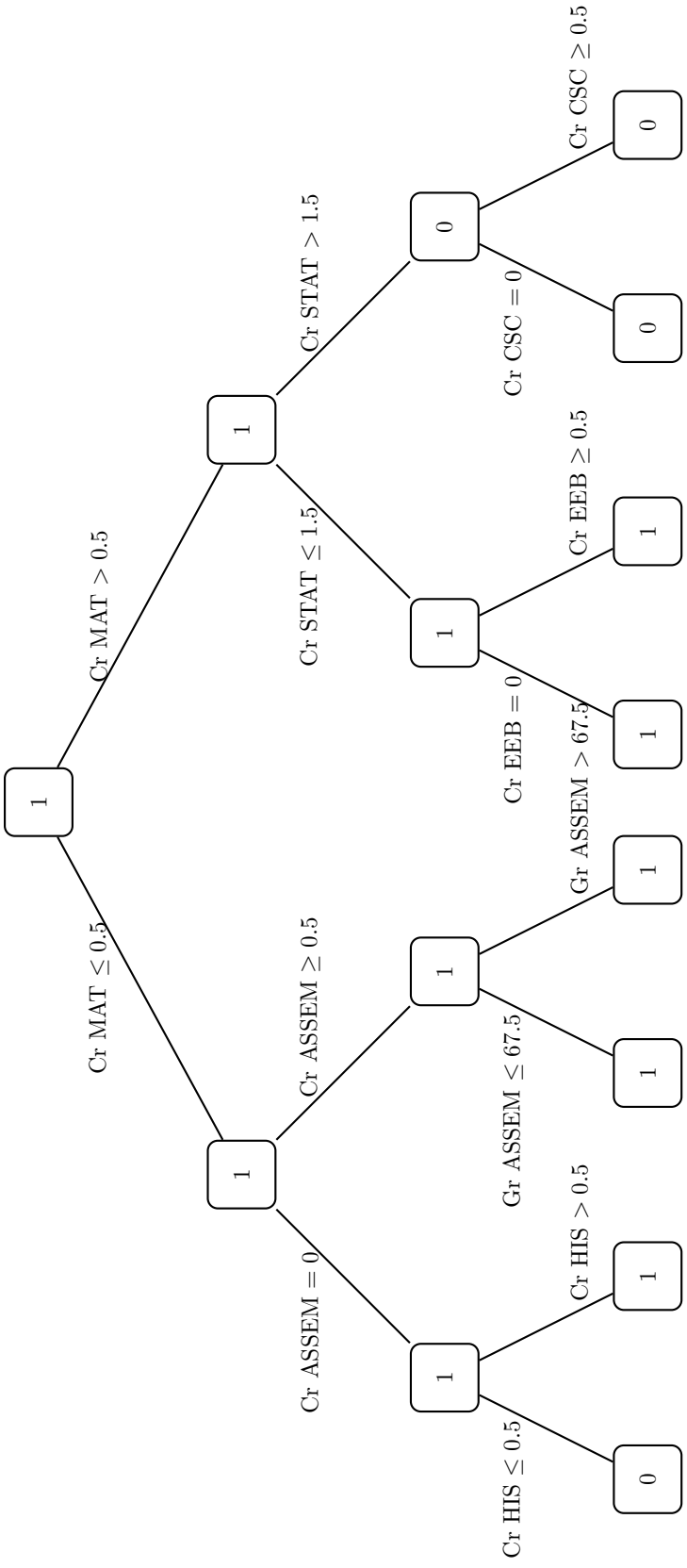


Figure 4.9: Upper part of a BEST decision tree when predicting program completion.

Chapter 5

Variational Autoencoders: theory and implementations

In this short chapter, we discuss the differences between the theoretical model and common implementations of VAEs. Those differences are there for a reason; they fix some problems that appear when implementing a VAE in its simplest form. First, we define the simple VAE and we provide a visualization of the problems this model suffers from. Second, we demonstrate empirically that common implementations manage to circumvent these problems. Third, we argue why these modifications empirically benefit the simple VAE based on the literature. Fourth, we demonstrate how these implementations do not respect the theory and why this is a problem. Finally, we propose potential solutions to those issues which would lead to an updated theory along with concordant implementations. However, those solutions have not been tested yet; we are demonstrating the existence of this gap between the theory and implementations to begin. In other words, only the first three steps of this project are completed at the moment of writing the thesis.

This chapter is not based on a published or submitted research paper yet; it is based on work that has begun a few years back and that is still ongoing. We are currently working on adapting the content of this chapter into publishable work. This chapter contains background information and observations we will refer to in the chapters that follow.

5.1 The simple variational autoencoder

In this section we define what we refer as the *simple VAE* in this chapter. It is actually the VAE model introduced as an example in section 2.2.3. We use this model to illustrate the difference between what was theoretically proposed and the common implementations.

The model is composed of a set of observed variables, which are identified as $\mathbf{x}$ and a set of unobserved latent variables, identified as $\mathbf{z}$. We assume $p(\mathbf{z})$ to be $N(0, I)$ and that $\mathbf{x}|z \sim N(\mu_x, \sigma_x)$. We also suppose that the dimension of $\mathbf{z}$ is d which is much lower than the dimension of $\mathbf{x}$, m.

Furthermore, in this model the parameters of the observed distribution $((\mu_x, \sigma_x))$ are continuous functions of the latent variable $\mathbf{z}$; $\theta = [\mu_x, \sigma_x] = f_x(z)$, to use a short notation, we identify $\mu_x(z)$ as the

function that takes z as input and return the parameters μ_x associated with this value and same for $\sigma_x(z)$ or simply $\theta(z) : \mathbb{R}^d \to \mathbb{R}^m \times \mathbb{R}^m_+$.

To ensure that this link function is as flexible as possible, a NN is used; $\theta(z)$ is a NN. It allows for a maximum amount of flexibility but in turn makes the posterior of the latent $p_\theta(\mathbf{z}|x)$ intractable and consequently the EM algorithm cannot be used. The proposed solution is to approximate $p_\theta(\mathbf{z}|x)$ with $q_\varphi(\mathbf{z}|x)$ a distribution of our choice.

For this simple variational autoencoder we will use a normal distribution: $\mathbf{z}|x \sim N(\mu_z, \sigma_z^2)$. Here again the parameters μ_z and σ_z are function of $\mathbf{x}$: $\varphi = [\mu_z, \sigma_z] = f_z(x)$ or $\varphi(x) : \mathbb{R}^m \to \mathbb{R}^d \times \mathbb{R}^d_+$. This function is once again a NN.

The simple VAE model is graphically represented as follows :

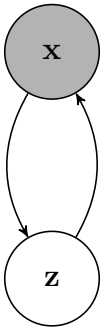
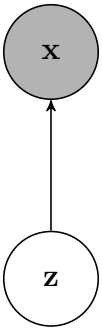


Figure 5.1: A graphical representation of the VAE architecture.

In Figure 5.1 the upward arrow represents the generative model (p) and the downward arrow represents the discriminative model (q) or inference network. Though not very useful now, it is practical for more complex model to separate both networks:



(a) The generative model assumes $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|z)$.

(b) The inference model. Given observations x we can infer the latent variable using $q_\varphi(\mathbf{z}|x)$.

5.1.1 Maximization of the ELBO

As previously mentioned in chapter 2, because it is impossible to compute the posterior distribution of the latent $p_\theta(\mathbf{z}|x)$ we cannot compute $\mathbf{E}_{p_\theta(\mathbf{z}|x)}[\ln p_\theta(\mathbf{z}, \mathbf{x})]$ and thus EM is not a viable solution here. The proposed solution is variational inference: we replace $p_\theta(\mathbf{z}|x)$ with an approximate distribution $q_\varphi(\mathbf{z}|x)$

and attempt to maximize the ELBO

$$\mathcal{L}(\varphi, \theta) = \mathbf{E}_{q_\varphi(\mathbf{z}|x)} \left[\ln p_\theta(\mathbf{z}) + \ln p_\theta(\mathbf{x}|z) - \ln q_\varphi(\mathbf{z}|x) \right].$$

The common strategy is to run a gradient-based optimizer on a Monte Carlo sample of the ELBO

$$\ln p_\theta(\mathbf{z}) + \ln p_\theta(\mathbf{x}|z) - \ln q_\varphi(\mathbf{z}|x) \quad z \sim q_\varphi(\mathbf{z}|x),$$

where we draw a new Monte Carlo sample at every steps of the optimization. To discuss further the current successful implementations, let us reorganize the terms in the ELBO

$$\begin{aligned} \mathcal{L}(\varphi, \theta) &= \mathbf{E}_{q_\varphi(\mathbf{z}|x)} \left[\ln p_\theta(\mathbf{z}) + \ln p_\theta(\mathbf{x}|z) - \ln q_\varphi(\mathbf{z}|x) \right] \\ &= \mathbf{E}_{q_\varphi(\mathbf{z}|x)} \left[\ln p_\theta(\mathbf{x}|z) - (\ln q_\varphi(\mathbf{z}|x) - \ln p_\theta(\mathbf{z})) \right] \\ &= \mathbf{E}_{q_\varphi(\mathbf{z}|x)} \left[\ln p_\theta(\mathbf{x}|z) \right] - \mathbf{E}_{q_\varphi(\mathbf{z}|x)} \left[\ln q_\varphi(\mathbf{z}|x) - \ln p_\theta(\mathbf{z}) \right] \\ &= \underbrace{\mathbf{E}_{q_\varphi(\mathbf{z}|x)} \left[\ln p_\theta(\mathbf{x}|z) \right]}_{\text{Reconstruction error}} - \underbrace{KL\left(q_\varphi(\mathbf{z}|x) | p_\theta(\mathbf{z}) \right)}_{\text{Regularization term}}. \end{aligned} \tag{5.1}$$

It is common to perceive the ELBO with respect to those two terms. We can see this as a regularized optimization problem where we want to maximize the first term, the observed-data likelihood, and where the second term works as penalization that discourages $q_\varphi(\mathbf{z}|x)$ from drifting far away from a $N(0, I)$ which represents the effect of the prior with a Bayesian flavour.

5.1.2 Practical uses

Dimensionality reduction and representation learning

A VAE is an unsupervised learning model like k-means clustering, GMM or Principal Component Analysis (PCA). Just like these other techniques, VAE can be used for dimensionality reduction. If the code z is a of much lower dimension, given the fitted encoding function φ and decoding function θ we can easily encode large observations x into the parameters of their lower-dimension representation z and also decode this representation to get the parameters of the reconstructed observation distribution. In contrast to k-means or PCA, VAE offers a probabilistic dimensionality reduction rather than a deterministic one which is similar to what GMM or pPCA offers in that aspect.

Dimensionality reduction is very useful for storage purposes or message transmission. A VAE can be directly used for lossy compression where z is the compressed representation x. VAEs can also be used as building blocks in more complex compression schemes, for instance Townsend et al. [145] constructed a lossless compression algorithm (BB-ANS) using VAEs. Besides, it is also quite common to apply supervised learning techniques to the latent representation itself, as discussed later.

Manifold learning is often a synonym of non-linear dimensionality reduction and in the context of machine learning is often viewed as a feature extraction step. Furthermore, the lower-dimension representation itself can be analysed sometimes [86, 114, 121, 83] to visualize the compression process. Since compression functions are continuous, it also allows us to better understand distances in the high-dimensional space $\mathcal{X}$.

Generator

A VAE is a generative model [83]. Indeed, since a prior distribution is assumed for the latent variable, $\mathbf{z} \sim N(0, I)$ we have a fully defined joint distribution $p_\theta(\mathbf{x}, \mathbf{z})$ and it is possible to generate new *observations* using ancestral sampling. Ancestral sampling [15] is the generative procedure for graphical models. The graphical representation of VAEs, a Bayesian network [88], represents a set of factorization and conditional independence assumptions which induces a natural sequence of events. In the simple VAE case the assumed factorization is :$p(x, z) = p(z)p(x|z)$. This leads to the following ancestral sampling procedure:

Algorithm 6: Ancestral Sampling with the simple VAE
INPUT: n desired size of the generated sample 1) Sample z from $N(0, I)$. 2) Process z through the NNs θ to get $\mu_x(z)$ and $\sigma_x(z)$. 3) Sample x from $N(\mu_x(z), \sigma_x(z))$. 4) Return x OUTPUT: a sample x of size n

Doing so allows us to generate new data points x according to its assumed distribution.

Semi-Supervised learning

VAEs can also be solutions to semi-supervised problems as was proposed by Kingma [85, 83] shortly after the release of the introductory paper [86]. Various semi-supervised VAE models have been proposed [85, 120, 107, 155]; and Rastgoufard [121] offers a thorough analysis of these models applied to various semi-supervised tasks. Semi-supervision aims at learning the *supervised relation* between the observations x and the labels y given a data set where multiple observations are not labelled. In other words, given a standard labelled data set can we improve the classifier by incorporating additional unlabelled observations ? It is quite an important problem since expert labelling is far more costly than the process of collecting raw data [121].

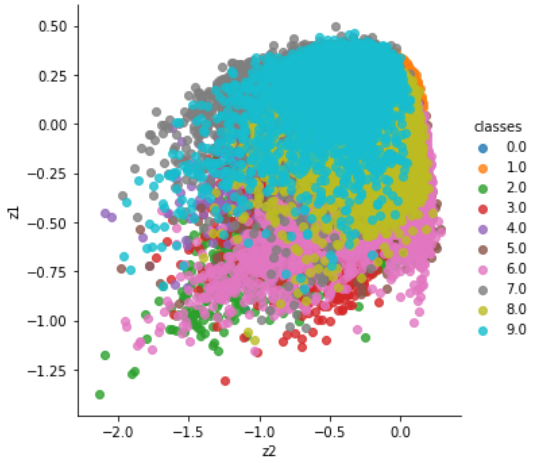
Observations from different classes are likely to cluster in different regions of the latent representation. In other words, observations x attached to different labels y will probably have different latent representation z. We can use the labelled observation to find an appropriate prediction function h that takes z as input and return a predicted label $\hat{y}$. The strategy behind semi-supervised learning is to leverage the large amount of unlabelled point to improve the encoding function $q_\varphi(\mathbf{z}|x)$ thus improving the classification mechanism.

In fact, the encoder q can be perceived as the feature-extraction step [153, 1, 122]; $\mathbf{z} \sim q(\mathbf{z}|x)$ is the vector of features extracted from the image x and it is easier to classify observations using those features than when using the original observations x. The classifier h is trained on the features z using labelled data set $S_l = \{(x_i, y_i)|i \in 1..n_l\}$ but we can use the additional unlabelled data $S_u = \{x_j|j \in 1...n_u\}$ to improve the feature extractor q.

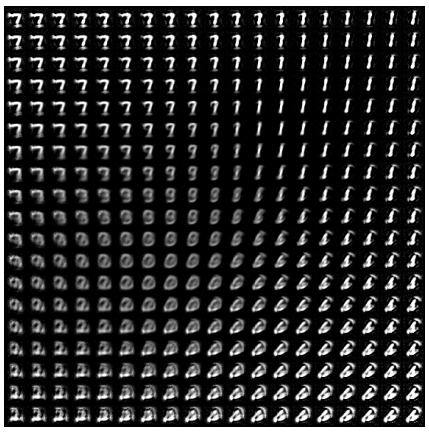
5.2 Visualization of the simple VAE

In this section we demonstrate problems when implementing the simple VAE directly. The VAE and it's associated generative procedures are implemented in Python and we will use the well-known MNIST data set [95] to visualize some of these problems. In hand-written document analysis, the MNIST data set introduced by LeCun & al. [95] quickly became a benchmark for hand-written digits recognition and is now a rite of passage for computer vision algorithms. It contains more than 60,000 images in shades of grey of hand-written digits of size 28 by 28 pixels.

The following images and plots are visual supports for our arguments about the simple VAE problems. We produce plots and images that illustrates how VAE performs in tasks mentioned before; compression and generation. Semi-supervised applications are left out for now, but VAEs limitations in semi-supervision problems has been extensively studied in Rastgoufard's thesis [121]. For compression we look at the latent space and its associated reconstruction $\mu_x(z)$, possible if $d = 2$, as well as an example of reconstruction to observe the *loss* incurred by the compression and decompression process.



(a) Observations x projected onto its latent representation using $z \sim N(\mu_z(x), \sigma_z(x))$.



(b) Decoded latent space using $\mu_x(z)$.

Figure 5.3: Latent space visualization of a simple VAE with latent space of dimension $d = 2$

Figure 5.4: Images x on the top row and its reconstruction $\mu_x(q_\varphi(x))$ on the bottom row produce from a simple VAE with latent space of dimension $d = 2$

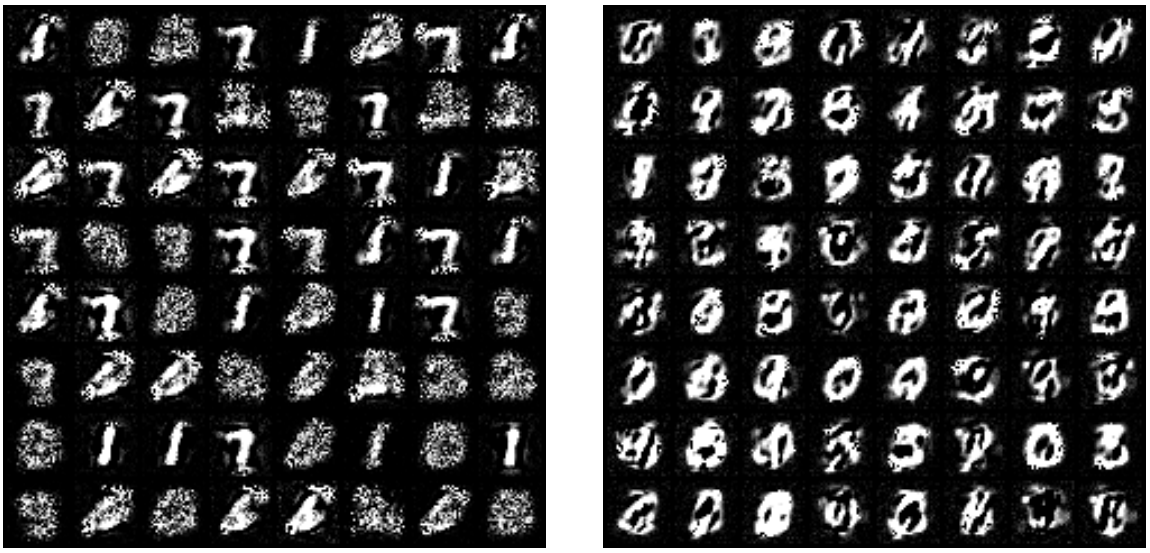
Figure 5.5: Images x on the top row and its reconstruction $\mu_x(q_\varphi(x))$ on the bottom row produce from a simple VAE with latent space of dimension $d = 20$

Figures 5.4 and 5.5 contain images and their associated reconstruction. We see multiple imperfections, blurry images and sometimes the reconstructed digit is a completely different digit. Of course, we do expect to lose some details when reducing the dimension from 784 to 20, but we know NNs allows for function complex enough and we hope to achieve better results than what is obtainable with PCA which compresses the data with linear combination.

Figure 5.6: Examples of reconstruction produced by PCA included in Bishop's book [15]. The image to the left is a real image and other images are reconstruction with latent space of size $d = 1$, $d = 10$, $d = 50$ and $d = 250$ respectively.

PCA uses simple linear combination for compression and decompression. However, it achieves reconstruction of similar quality with a latent space of size $d = 10$ (third image of Figure 5.6) than the simple VAE with a latent space of size $d = 20$ who relies on NN as for compression and decompression. This is a disappointment.

For generation, we use ancestral sampling to produce a sample of 64 images:

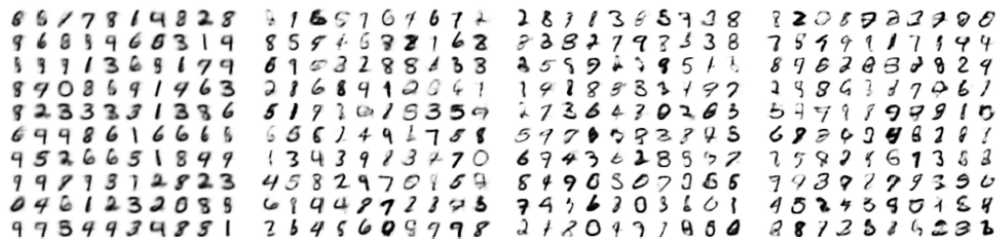


(a) $d = 2$ (b) $d = 20$

Figure 5.7: Sample obtained from the ancestral sampling described in the previous section.

Now we observe another major problem; the images generated are blurry, contains multiple imperfection and lack diversity. A human eye would judge harshly those images; they do not look like realistic hand-written digits. As it stands, with an exact implementation of the simple VAE, the compression

and reconstruction abilities of such models are equivalent to PCA and the generated images are not impressive. None of these problems are mentioned in the papers that originally presented this model [86, 73]. For instance, here are the samples available in Kingma's thesis [83] :



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure 5.8: Snapshot of the result section of Kingma's thesis.

It is unlikely that neither of the first authors did not encounter any of these problems. By avoiding this discussion, it falls on the users to figure out how to implement the needed modifications and this prevents the model to be used reliably on real-data problems as is.

5.3 Algorithmic solutions

As we observed in the previous section, a direct implementation of the simple VAE proposed in the literature suffers from problems for both reconstruction and image generation. The figures of section 5.2 reveal some problems of the simple VAE. However figures much more flattering were published in articles discussed above. To produce those images, some important modifications were done by researchers *under the hood* of the proposed VAE of section 2.2.3, we will discuss those modifications in their respective subsection.

In this section we explore successful implementations of VAEs and we highlight the differences between the simple VAE proposed and the common implementations. We discuss these differences and their impact on the resulting model; how they fix some problem but drastically steer the model away from its original proposed form. We named this section *algorithmic solution* since the explored modification are algorithmic rather than theoretical. Our main objective in this chapter is to raise awareness and motivate further research in this area.

5.3.1 Tradeoff between reconstruction and regularization

Remember that

$$\begin{aligned}\mathcal{L}(\varphi,\theta) &= \mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(z) + \ln p_\theta(x|z) - \ln q_\varphi(z|x)\right] \\ &= \underbrace{\mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(x|z)\right]}_{\text{Reconstruction error}} - \underbrace{KL\left(q_\varphi(z|x)|p_\theta(z)\right)}_{\text{Regularization term}}. \qquad (5.2)\end{aligned}$$

(5.3)

Figure 5.9: Images x on the top row and its reconstruction $\mu_x(q_\varphi(x))$ on the bottom row produced with a β-VAE with latent space of dimension $d = 20$

In multiple implementations we observed a wide range of modifications to the objective function where both the reconstruction error and the regularization term are considered separately. To begin we will address the balance, or lack thereof, between the two components of the ELBO.

When perceiving the ELBO as a regularized optimization problem as defined in section 5.1.1, the need for an hyper-parameter controlling the strength of the regularization might seem beneficial. It is now common to add a hyper-parameter, say β in the objective function to allow us to control the balance between the reconstruction error and the regularization

$$\mathbf{E}_{q_\varphi(z|x)}\left[\ln p_\theta(x|z)\right] - \beta KL\left(q_\varphi(z|x)|p_\theta(z)\right). \tag{5.4}$$

Models optimized with the objective function of Equation 5.4 are known as β-VAEs [63, 24] and they were developed to improve the disentanglement of the latent representation. In fact, their ability to form a disentangled representation has been well studied [63, 24]. However, there is little discussion on the effect of this hyper-parameter on the generative abilities of β-VAEs and how to select the hyper-parameter β appropriately.

The authors indicate that a large β is *putting a stronger constraint on the latent bottleneck than in the original VAE formulation which... should encourage the model to learn the most efficient representation of the data.* They also claim that the regularizing term in the objective function encourages conditional independence in q_φ. However this is done to the detriment of the reconstruction term and to the detriment of variability in generated samples.

Simply put, Equation 5.4 directly implies that small β leads to a more accurate reconstruction and large β to more regularization. In other words, small β enables the algorithm to compress the data to a lower-dimensional space and reconstruct an almost perfect image:

In Figure 5.9 we observe much better reconstruction that previously in Figure 5.5.

5.3.2 Reconstruction term

Secondly, let us discuss the implementation of the reconstruction term. If we assume that $\mathbf{x}|z \sim N$ then

$$\begin{aligned}\ln p_\theta(x|z) &= \ln\left(\frac{1}{\sqrt{2\pi\sigma(z)^2}}\exp\left(\frac{-(x-\mu(z))^2}{2\sigma(z)^2}\right)\right)\\ &= -\frac{1}{2}\ln\left(2\pi\sigma(z)^2\right) - \frac{(x-\mu(z))^2}{2\sigma(z)^2}.\end{aligned} \tag{5.5}$$

Common implementations do not maximize the reconstruction term of Equation 5.5. Instead the NN θ returns an output of the same size as x and minimize the mean squared error (MSE) between x and

the reconstructed $\bar{x} = \mu(z)$. The motivation is that minimizing the MSE is equivalent to maximizing the log-likelihood for a normal distribution with a fixed $\sigma_x = 1$ (I)

$$-\frac{1}{2}\ln(2\pi) - \frac{(x-\mu(z))^2}{2} \propto -(x-\mu(z))^2.$$

Based on empirical result fixing $\sigma(z)$ produces better reconstructed images:

Figure 5.10: Images x on the top row and its reconstruction $\mu_x(q_\varphi(x))$ on the bottom row produce from a simple VAE with latent space of dimension $d = 2$ and $\sigma = 1$

Figure 5.11: Images x on the top row and its reconstruction $\mu_x(q_\varphi(x))$ on the bottom row produce from a simple VAE with latent space of dimension $d = 20$ and $\sigma = 1$

In Figures5.10 and 5.11 we have noticeably better reconstruction than in Figures 5.4 and 5.5.

5.3.3 Modification to the ancestral sampling procedure

Finally, let us introduce a modification done to the data generation technique. We previously discussed how VAEs were presented as generative models and that the graphical representation suggested the Ancestral Sampling technique detailed in Algorithm 6.

However, all the implementations found online, including Kingma's implementation [85], do not rely on ancestral sampling. Actually, none of the implementation we found sample images from $p(x|z)$, instead x is deterministic given z, which is why we coined this technique *deterministic sampling*.

Algorithm 7: Deterministic Sampling
INPUT: n desired size of the generated sample 1) Sample z from $N(0, I)$. 2) Process z through the NNs θ to get $\mu_x(z)$ and $\sigma_x(z)$. 3) Return $\mu_x(z)$ OUTPUT: a sample of means (μ_x) of size n

The only difference between the samples of Figures 5.7 and Figures 5.12 is the sampling technique. In other words, we have trained a simple VAE, as introduced in section 5.1, but instead of generating x from $p_\theta(\mathbf{x}|z)$, we simply returned $\mu_x(z)$. The difference in the quality is noticeable on eyesight.

(a) $d = 2$ (b) $d = 20$

Figure 5.12: Sample obtained from $\mu_x(z)$ where $z \sim N(0, I)$.

Figure 5.13: Images x on the top row and its reconstruction $\mu_x(q_\varphi(x))$ on the bottom row produced from a simple VAE with latent space of dimension $d = 2$ and $\sigma = 0.0001$

Figure 5.14: Images x on the top row and its reconstruction $\mu_x(q_\varphi(x))$ on the bottom row produced from a simple VAE with latent space of dimension $d = 20$ and $\sigma = 0.0001$

5.3.4 Effect on the model optimized

The three algorithmic solutions discussed above have one thing in common; they all directly affect how the total observed variance is distributed in the resulting model.

For β-VAE, the β parameter influences the variance of $q_\varphi(\mathbf{z}|x)$. Large β pushed $q_\varphi(\mathbf{z}|x)$ towards a $N(0, I)$ distribution while small β allows $q_\varphi(\mathbf{z}|x)$ having a much large variance with correlated dimensions.

Similarly, when fixing $\sigma_x = I$, the distribution of the total variability is now fixed by the user. Fixing σ_x constrain the variance of $p_\theta(\mathbf{x}|z)$. Like β can be adjusted to distribute the variability in different way, σ_x can be fixed to different values. For instance, if we wanted to force the latent variable to take on larger portions of the variability we could fix σ_x to a small value, this was also suggested by Lucas et al. [105].

Figure 5.14 shows impressive reconstruction. These images are, on eyesight, as good as those of Figure 5.9 but using totally different approaches. Earlier with selected a small β thus relaxing the constraint applied by the regularization term and here we have fixed a small σ_x. Those two techniques allow for most of the variability to be explained by the latent space. The fact that adjusting σ_x leads to similar fit than adjusting β was mentioned by Lucas [105] but it was not mentioned that this is due to both having similar effect on how the total variability is distributed across both components, latent and observed, of the VAE.

Finally, the last modification discussed constrains the variance of $p_\theta(\mathbf{x}|z)$ when generating. For a fixed z then $x = \mu_x(z)$ which is equivalent to fixing $\sigma_x = 0$; this turns the generating distribution from a Normal to a Dirac Delta distribution.

Another way to perceive this constraint is that it ensures a correlation between the pixels. When using $\mu_x(z)$ every pixel has the exact same distribution value of 0.5. This is because the mean of a normal distribution is also its median; $F(\mu) = 0.5$ where F is the distribution function of any normal distribution. Consequently, pixels are perfectly correlated in their distribution value. We can produce images that look just as good by sampling $z \sim N(0, I)$ and then outputting $\mu_x(z) + \alpha \times \sigma_x(z)$ for any α as seen in Figure 5.15.

(a) $\mu_x(z) - 0.1 \times \sigma_x(z)$ (b) $\mu_x(z) + 0.1 \times \sigma_x(z)$

Figure 5.15: Samples obtained from a simple VAE with $z \sim N(0, I)$.

To summarize, these three algorithmic solutions improve either the compression/decompression abilities or generative abilities of the simple VAE by putting constraints on the variance either by modifying the objective function while training or by modifying the sampling procedure when generating new observations. In the next section we argue that these modifications create new problems and we demonstrate that the model resulting from those modifications does not respect the theory any longer which, in turn, makes these new problems hard to solve.

5.4 Issues with algorithmic solution

5.4.1 Application issues

The common algorithmic solutions discussed above solve some issues of the simple VAE as illustrated in section 5.3, however, new problems also appear.

First, selecting the β parameter is a complicated task where the user has to define how important is the reconstruction relatively to the regularization of q_φ. To this day there is no automated way to select the right value for β. Similarly, if we desire to fix σ_x the value of this fixed variance has to be established heuristically.

Second, the improvement in reconstruction observed when fixing a small β or a small σ_x comes to the detriment of the generative abilities of VAE. In fact, small β or small fixed σ_x leads to a q_φ with high variance as explained earlier. This is problematic from a generative perspective. Remember that we optimize a Monte Carlo sample of the ELBO, thus we train $p_\theta(\mathbf{x}|z)$ using zs sampled from $q_\varphi(\mathbf{z}|x)$. In other words, the NN function θ is trained with zs generated from $q_\varphi(\mathbf{z}|x)$. Consequently if $q_\varphi(\mathbf{z})$ and $p_\theta(\mathbf{z})$ have drastically different supports then we do not know how does the NN θ will react when receiving inputs z sampled from $p_\theta(\mathbf{z})$.

Another generative problem arises with large β, this leads to a lack of variability in generated images. The lack of variability in the generated data happens when $q_\varphi(\mathbf{z}|x)$ resembles too much the prior $p_\theta(\mathbf{z})$, instead of getting close to the intractable posterior $p_\theta(\mathbf{z}|x)$, and this problem has been recently coined *posterior collapse*. As a matter of fact, if $q_\varphi(\mathbf{z}|x) \approx p_\theta(\mathbf{z})$ then $q_\varphi(\mathbf{z}|x)$ is *independent* of x; $q_\varphi(\mathbf{z}|x)$ does not vary as x varies. This is not intended; we want the latent representation of x obtained through $q_\varphi(\mathbf{z}|x)$ to contain information about x and thus to be different for different xs. This leads to a latent space that does not contain information about the observed space and this leads to an homogenized reconstruction.

Figure 5.16 provides a visualization of the problems caused when β is either too large or too small. In Figure 5.16a we see the high variance latent space comparatively to the much more constraint counterpart of Figure 5.16c. The effect of posterior collapse, with too large β, can also be observed in Figure 5.16d where $\mu_x(z)$ is constant in z and the resulting image is the average of all digits.

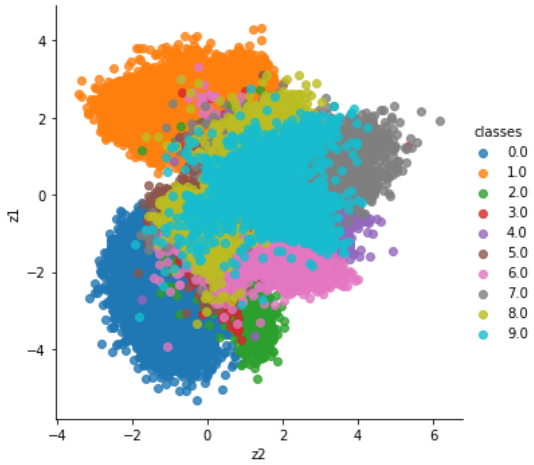
These problems are easier to observe when using VAEs on a single digit data set:

We see bigger variability in the images spanned by the latent space in Figure 5.17b but also more imperfections. This happens when θ has to process zs unobserved in the training process. This contrast with Figure 5.17d where all images are relatively good but they all look alike. This is a symptom of posterior collapse.

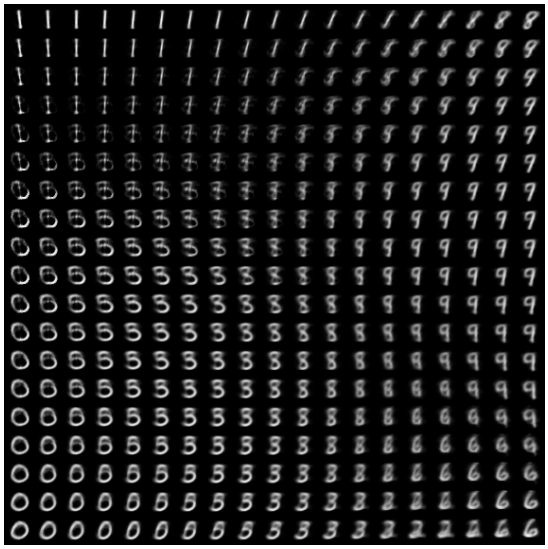
Similarly we have high variance for $q_\varphi(\mathbf{z}|x)$ when fixing σ_x to a small value such as observed in Figure 5.18.

5.4.2 Theoretical issues

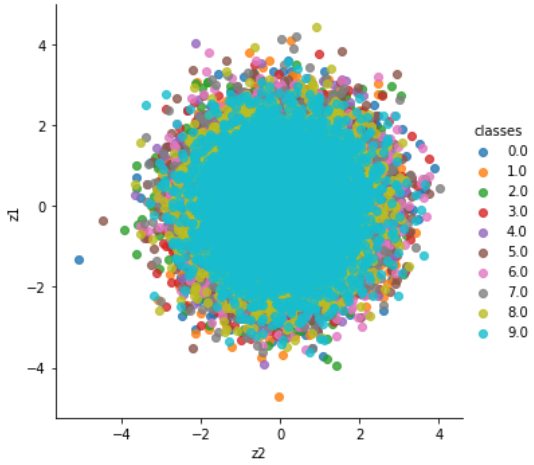
Additionally, we want to raise awareness towards theoretical issues with the algorithmic solutions detailed in the previous section.



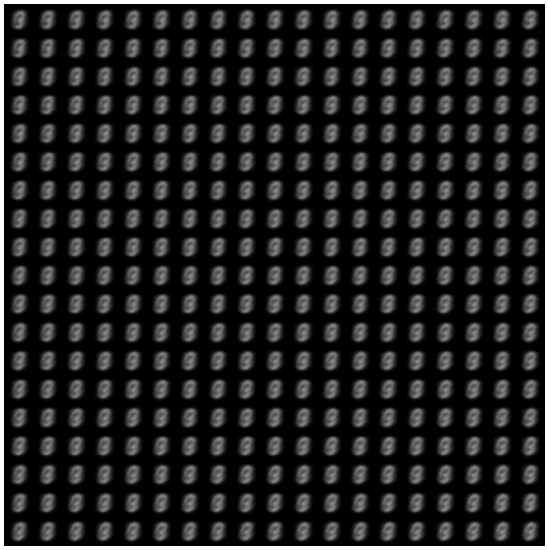
(a) Observations x projected onto its latent representation using $z \sim N(\mu_z(x), \sigma_z(x))$ with small β



(b) Decoded latent space using $\mu_x(z)$ with small β

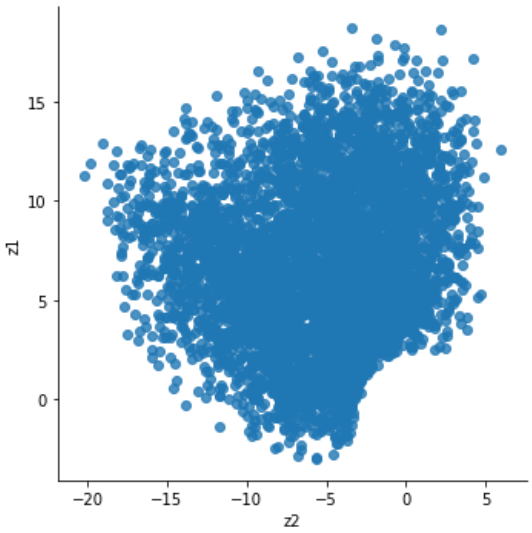


(c) Observations x projected onto its latent representation using $z \sim N(\mu_z(x), \sigma_z(x))$ with large β

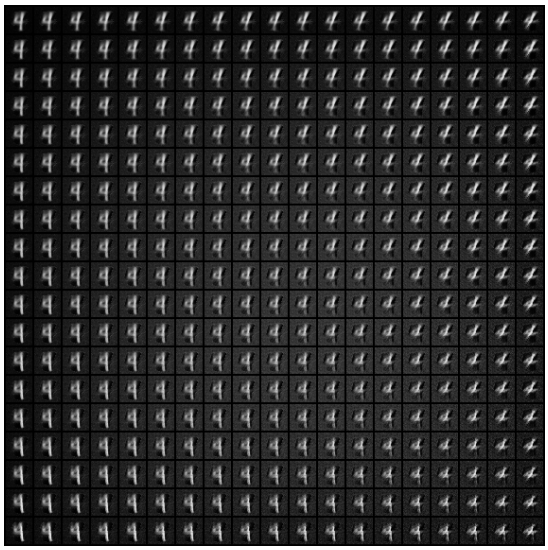


(d) Decoded latent space using $\mu_x(z)$ with large β.

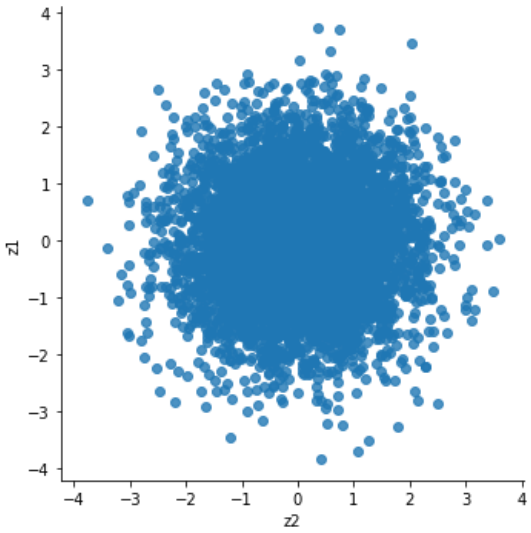
Figure 5.16: Visualization of the latent representation.



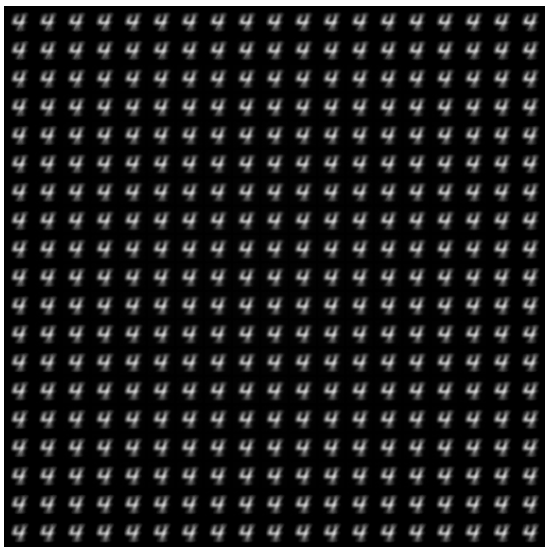
(a) Observations x projected onto its latent representation using $z \sim N(\mu_z(x), \sigma_z(x))$ with small β.



(b) Decoded latent space using $\mu_x(z)$ small β.

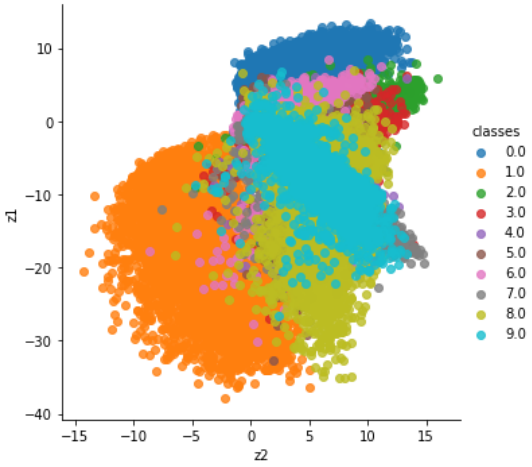


(c) Observations x projected onto its latent representation using $z \sim N(\mu_z(x), \sigma_z(x))$ with large β

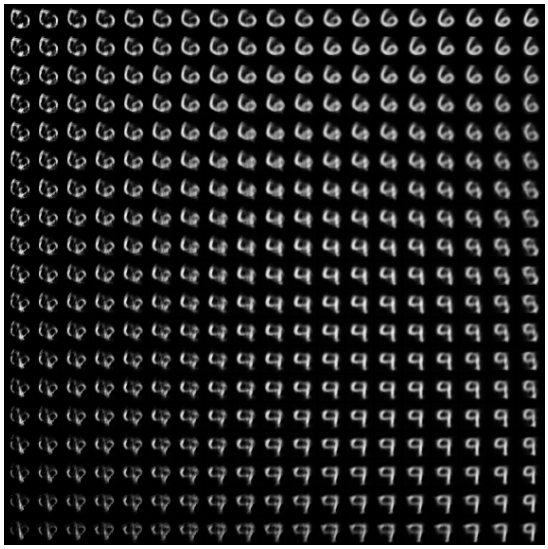


(d) Decoded latent space using $\mu_x(z)$ with large β

Figure 5.17: Visualization of the latent representation.



(a) Posterior distribution of the latent : $p(z|x)$



(b) Decoded latent space using $\mu_x(z)$.

Figure 5.18: VAE with latent space of dimension $d = 2$ and $\sigma = 0.0001$

To begin, we quickly address the choice of observed data distribution. Though the VAE model is very flexible in the form $p_\theta(\mathbf{x}|z)$ can take, lots of implementations use the Bernoulli distribution [86, 137, 91, 40]. This is a problem since the Bernoulli distribution supports binary variable and thus it should not be used to model pixels continuously distributed in $(0, 1)$. Similarly, we also encounter a support problem with normal distribution. Pixels are continuously distributed in $(0, 1)$ but the Normal support is infinite. However these are more data compatibility problems rather than theoretical issues with the proposed algorithmic solutions.

When strictly considering the solutions discussed, the biggest problem is the violation of certain theoretical properties and guarantees of the simple VAE. For the β-VAE, by selecting a $\beta < 1$ the resulting objective function is no longer a lower bound of the marginal log-likelihood $\log p(x)$ and thus we are losing an important theoretical guarantee of the model.

We can take the β-VAE concept to its limit and fix $\beta = 0$, this produces the best reconstruction possible but also eliminates one of the novelties of VAEs; the distribution of the latent variable z. In fact, when $\beta = 0$ the parameters of $q_\varphi(\mathbf{z}|x)$ are not estimated anymore. The resulting model is much closer to an AE fitted by maximizing a likelihood function.

A similar problem comes with combining the use of the MSE as the reconstruction error and the deterministic sampling procedure. If we optimize $\mu_x(z)$ by minimizing the MSE, thus fixing σ_x when training and the data generated is $\mu_x(z)$ itself, thus fixing σ_x when generating then we got rid of the probabilistic components of x. Indeed, the θ parameters can be reduced to μ_x and the variance of $p_\theta(\mathbf{x}|z)$ is not considered at any point in time during training nor generation. In other words, the resulting model is totally deterministic in x given z.

Combining these modifications altogether, and taking the β-VAE to its extreme case we now have

the following objective function:

$$(x - \mu_x(z))^2 \quad \text{where } z \sim q_\varphi(\mathbf{z}|x) \tag{5.6}$$

When maximizing the resulting objective function of Equation 5.6, only $\mu_x(z)$ is trained. The model is now an AE with a NN decoder optimized by minimizing the mean-squared reconstruction error and an untrained probabilistic encoder.

5.5 Future work

As explained in section 5.3.4 the algorithmic solutions have one thing in common: they all influence how the total variance is distributed between the latent variables and observed variables. Hence we believe the cause of the problems observed in section 5.2 with the simple VAE is the lack of identifiability between the variability attributed to both the observed and the latent component. In short, when fitting latent variable models the total variability within the observed data x is split between the variance of the latent variable and the variance of the observed variable and there exist infinitely many ways to split the total variance in two. To solve this identifiability problem it is common to fix the variance of one of the components or to decide how to distribute the variability between the two components when establishing the optimization procedure.

For instance, in PCA the latent representation is designed to take on as much variability from the observed data space as possible. In PCA, the variance of the latent representation z of size d is the average of the d largest eigenvalue of the covariance matrix of the observed data.

We want to propose a new theoretical formulation, along with concordant implementation that solves this variance identifiability problem. We believe having implementations concordant to the theory is beneficial as it helps to generalize the model to new applications and it will allow us to rely on the theory if problems come up, which is not the case with the modification now that the resulting model has strayed away from the theoretical formulation. We also believe that fixing this variance identifiability problem would be beneficial as it would better grasp the natural variability in a wide range of data sets which is ignored in a deterministic AE.

Our goal is to allow for the total variance to be expressed differently than it is right now. We want to take a closer look at probabilistic PCA (pPCA) [144, 143]. In this model, the variance of the latent representation z of size d is the average of the d largest eigenvalue again and the variance of $p_\theta(x|z)$ is the average of the leftover eigenvalues. Compared to PCA where the maximum variance projection is enforced by the model, the solution in pPCA happens naturally without specific constraints. This is a strong result we hope to use in our future work on VAE in order to balance naturally both components' variance. Based on the results of PCA and pPCA where a maximum amount of the variability is attributed to the latent variable and on the results we observed with β-VAE it seems like for the simple VAE to produce better reconstruction and generations it needs to shift some of the observed distribution variance to the latent representation variance. Similarly, we could study GMMs, or any other model-based clustering. Studying the similarities and differences between VAEs and GMMs could provide us with interesting insight regarding the problems that VAEs suffer from.

Additionally, once this balance is fixed, we believe it is important that the observed distribution variance can be expressed in a more complete manner; we want to drop the conditional independence assumption.

$$\mathrm{Var}(\sum_{j=1}^{m} \mathbf{x}_j) = \sum_{j=1}^{m} \mathrm{Var}(\mathbf{x}_j) + \sum_{j \neq i}^{m} \mathrm{Cov}(\mathbf{x}_j, \mathbf{x}_i) \tag{5.7}$$

In the simple VAE, the normal distribution that models observations has a diagonal covariance matrix, which bottlenecks all of the observed distribution variance on the diagonal as suggested by the simple decomposition of the total variance of Equation 5.7.

In other words, even if the total variance was distributed optimally between $\mathbf{z}$ and $\mathbf{x}|z$ we would also need to let some of the covariance term of Equations 5.7 to be non-zero otherwise it will result in high individual variable variance. This should also better model real data such as images where pixels in a neighbourhood are highly correlated. In order to optimally fit a covariance matrix we are currently exploring ideas of spatial statistics.

5.6 Related literature

We faced those problems within the first few years of VAE's existence in early 2016 and slowly started working on this chapter. Back then, none of the literature available mentioned those problems neither how the small coding tricks established earlier were actually drastically changing the model.

Based on our research, the *posterior collapse* problem is the problem addressed the most in the literature and it is now fully recognized as a problem and received a lot of attention in the last few years. Though it was not in an attempt to solve the issue the paper presenting the β-VAE formulation [63, 24] was among the first to discuss the effect of the regularization term of the ELBO and it's potential effect of the variability in the images it produces. He et al. [62] recently provided an insightful investigation of posterior collapse; they suggest that the cause of posterior collapse is the inference of the approximate distribution lagging behind the true posterior at the early stages of training.

Alemi et al. [3] directly discussed the posterior collapsed problem with an information theory approach. The problem is indeed that z does not contain enough *information* about x and they propose to optimize VAEs in a way that maximizes the mutual information between the observed variables and the latent variables. Not only did they address the problem but they also encouraged research in that regard.

After a publication from Dai et al. [37] discussing the relationship between PCA and VAEs, Lucas et al. [105] made connections with the pPCA model. They demonstrate that the regularization term is only partially responsible for posterior collapse but mostly that the variance parameter of the decoding distribution was playing a huge role. This confirms what we suspect. The authors make a thorough analysis of the effect of the variance term and suggest that the optimization procedure naturally favours too much observed-data variance and suggest way to reduce it to solve the posterior collapse problem. Overall this paper is a great contribution towards solving some of the VAE issues.

Lucas et al. [105] also show that, for a linear VAE, the ELBO has the same global maximum as the log likelihood and thus the solution has scaled principal components as the columns of the decoder network.

They also show that using the ELBO objective does not introduce new local maxima. Finally, after establishing a metric for posterior collapse they demonstrate how fixing a small σ_x makes the posterior collapse problem completely disappear. Additionally, we have shown in this chapter that restraining the variance in $p(x|z)$ drastically improves the reconstruction abilities of VAEs. Now, many solutions and formulations have been proposed [4, 151, 128]recently to discuss the variance problems and we are excited to see such keen interest towards this problem.

Although we have only noticed few articles discussing the issue with the current generative problem, Dorta et al. [42, 41] came up with a similar observation that we did; $\mu_x(z)$ is commonly used to generate images because of the poor performances of ancestral sampling which is caused by the lack of correlation between pixels. The solution they proposed is a fully parametrizes covariance matrix for $p(x|z)$.

5.7 Conclusion

The VAE model as defined in the literature [86, 83] is built upon a rigorous theory and the described model is both innovative and a big contribution to the fields of machine learning and statistics alike. It extends latent variable models to allow for more flexible functions between the latent and the observation space and has empirically performed well on some real-data problems.

However it seems there are big difference between the theory and the popular implementations. The current implementations fix some of the problems encountered when using VAEs but they do so by taking out the components that made VAEs special. In this chapter, we demonstrated how most of the simple fixes we found online are progressively transforming a VAE into an AE. We demonstrated that these fixes also come with new problems. Finally, we provided a taste of the solutions we are currently working on.

Chapter 6

An evaluation of machine learning techniques in survival analysis

Predicting the outcome for children treated for Hodgkin-Lymphoma

In this chapter we analyse a data set containing information on children with Hodgkin Lymphoma (HL) enrolled in a clinical trial. Treatments received and survival status were collected together with other covariates such as demographics and clinical measurements. Our main task is to explore the potential of machine learning (ML) algorithms in a survival analysis context in order to improve over the Cox Proportional Hazard (CoxPH) model. We discuss the weaknesses of the CoxPH model we want to improve upon and then we introduce multiple algorithms, from well-established ones to more recent models, that solve these issues. We then compare every model according to the concordance index (c-index) and the Brier score. Finally, we produce a series of recommendations, based on this experiment, for practitioners of the medical field who would like to benefit from the recent advances in artificial intelligence.

In this chapter, the contributions are (1) a new survival analysis algorithm built upon the VAE architecture (SAVAE), (2) a thorough comparison of a wide range of survival analysis machine learning algorithms on a real data set and (3) a list of recommendations and a discussion regarding machine learning abilities to deal with real life medical data sets. The main contributions of this chapter were introduced in a workshop paper published at NeurIPS [12] and in a research article published in *Applied Artificial Intelligence* [13].

6.1 Introduction

There is increasing effort in medical research to applying ML algorithms to improve treatment decisions and predict patient outcomes. In this chapter, we explore the potential of ML algorithms to predict the outcome of children treated for Hodgkin-Lymphoma. As we want to minimize the side effect of intensive chemotherapy or radiation therapy, a major clinical concern is how, for a given patient, can we select a

treatment that eradicates the disease while keeping the intensity of the treatment, and the implied side effect, to a minimum.

In this chapter we introduce multiple ML algorithms adapted to our needs and compare them with the Cox proportional hazard model. As it is the case with many data set within this field, the response variable, time until death or relapse, was right-censored for patients without events and the data set is of relatively small size (n=1712). These characteristics are uncommon in typical ML applications and this is part of the challenge. In many cases, ML techniques are not designed to deal with censored observations and thus it restricts the techniques we can include in our case study. Medical data sets usually have a smaller number of observations than the data sets ML algorithms are tested on and consequently our results may differ from the results previously published in the ML literature.

We introduce the data set in section 2. In section 3 we introduce the algorithms tested. Then, in section 4 we present in detail the algorithm we developed. In section 5, we discuss the experimental set up and our results. Finally, in section 6, we hold a thorough discussion on the results, recommend further improvements and introduce open questions.

6.2 Data set

We have a data set of 1 712 patients, treated on the Children's Oncology Group trial AHOD0031, one of the largest trial of pediatric HL ever conducted. Each observation represents a patient suffering from Hodgkin Lymphoma. For every patient, characteristics and symptoms have been collected as well as the treatment selected by the physician for a total of 21 predictors. A table containing information on the predictors is in the appendix. The response is a time-to-event variable registered in number of days. We consider events to be either death or relapse. For patients without events, the response variable was right-censored at time of last seen, which is a well-known data structure in survival analysis. This data set and the data collecting technique are presented in detail by Friedman & al. [48] who previously analysed the same data set for other purposes.

6.3 Survival Analysis models

6.3.1 Benchmark : Cox Proportional Hazard Model

The Cox Proportional Hazard (CoxPH) model [34] serves as our benchmark model. It is widely used in medical sciences since it is robust, easy to use and produce highly interpretable results. It is a semi-parametric model that fits the hazard function, which represents the instantaneous rate of occurrence for the event of interest, using a partial likelihood function [35].

The CoxPH model fits the hazard function which contains two parts, a baseline hazard function of the time and a feature component which is a linear function of the predictors. The proportional hazard assumption assumes the time component and the feature component of the hazard function are proportional. In other words, the effect of the features is fixed through time. In the CoxPH model, the baseline hazard, which contains the time component, is usually unspecified so we cannot use the model directly to compute the hazard or to predict the survival function for a given set of covariates.

The main goal of this analysis is to test whether or not new ML models can outperform the CoxPH model. As recently developed ML models have shown great potential in many data analysis applications, it seems important to test their potential in the medical field. We selected models that improve upon at least one of the three following problems that are intrinsic to the CoxPH model. Problem (1): the proportional hazard assumption; we want models that allow for features effect to vary through time. Problem (2): the unspecified baseline hazard function; we want models able to predict the survival function itself. Problem (3): the linear combination of features; we want models that are able to grasp high order of interaction between the variable or non-linear combinations of the features.

Weibull regression: an additional benchmark

As recommended by our thesis committee, we incorporated a Weibull regression model [77] as a secondary benchmark model. This additional benchmark is closer in spirit than CoxPH to some of the new ML models, such as the model we propose, the SAVAE. Consequently, the Weibull regression is used as an additional benchmark.

The Weibull distribution is a continuous parametric failure time model. It extends the exponential distribution since it allows the hazard to vary through time. It fixes problem (2) because it does fit the distribution of the time-to-even completely however it is a member of the proportional hazard family; the effect of the features on the hazard is fixed through time. This model will be fit using the survival [140] R-package.

6.3.2 Conventional statistical learning models

Regression models

The first model to be tested is a member of the CoxPH family. One way to capture interactions between predictors in linear models, and thus improve towards problem (3), is to include interaction terms. Since typical medical data set contains few observations and many predictors, including all interactions usually leads to a saturated model.

To deal with this issue we use a variable selection model. Cox-Net [135] is an extension of the now well-know lasso regression [61] implemented in the glmnet package [49] and is the first model we experiment with. The Cox-Net is a lasso regression-style model that shrinks some coefficients of the model to zero and thus ensure the model is not saturated. The resulting model is as interpretable as the benchmark CoxPH model, but Cox-Net allows us to include all interactions in the base model without losing too many degrees of freedom.

Another approach based on regression models is the Multi-Task Logistic Regression (MTLR). Yu et al. [154] proposed the MTLR model which quickly became a benchmark in the ML community for survival analysis and was cited by many authors [106, 46, 156, 74]. The proposed technique directly models the survival function by combining multiple local logistic regression models and considers the dependency of these models. By modelling the survival distribution with a sequence of dependent logistic regression, this model captures time-varying effects of features and thus the proportional hazard assumption is not needed. The model also grants the ability to predict survival time for individual

patients. This model solves both problem (1) and (2). For our case study, we use the MTLR R-package [58] recently implemented by Haider.

Survival tree models

Decision trees [22] and random forests [18, 20] are known for their ability to detect and naturally incorporate high degrees of interactions among the predictors which is helpful towards problem (3). They are well-established models and they also make very little assumption about the data set. For these reasons, this family of models is a natural choice for our case study.

Multiple adaptations of decision trees were suggested for survival analysis and are commonly referred as survival trees. The idea suggested by many authors is to modify the splitting criteria of decision trees to accommodate for right-censored data. Based on previously published reviews of survival trees [92, 16], we have selected four techniques for the case study.

One of the oldest survival tree models that was implemented in R is the Relative Risk Survival Tree [93]. This survival tree algorithm uses most of the architecture established by CART [22] but also borrows ideas from the CoxPH model. The model suggested by LeBlanc et al. assumes proportional hazards and partitions the data to maximize the difference in relative risk between regions. This technique is implemented in the rpart R-package [141].

We also selected a few ensemble methods. To begin, Hothorn et al. [67] proposed a new technique to aggregate survival decision trees that can produce conditional survival function, which solves problem (2). To predict the survival probabilities of a new observation, they use an ensemble of survival trees [93] to determine a set of observations similar to the one in need of a prediction. They then use this set of observations to generate the Kaplan-Meier estimates for the new one. Their proposed technique is available in the ipred R-package [116]. A year later, Hothorn et al. [64, 139] proposed a new ensemble technique able to produce log-survival time estimates instead. We experiment with this technique with the implementation available in the party R-package [66, 65].

Finally, the latest development in random forests for survival analysis is Random Survival Forests [70]. This implementation of a random survival forest was shown to be consistent [69] and it comes with high-dimensional variable selection tools [71]. This model was implemented in the randomForestSRC R-package [72].

6.3.3 Newly established models

Deep learning models

The first new model we experiment with is built upon the most popular architecture of models in recent years: deep neural networks. Yu et al. [154] MTLR model inspired many modifications [106, 46, 156, 74] in order to include a deep-learning component to the model. The main purpose is to allow for interactions and non-linear effect of the predictors. For example, Fotso [46, 47] suggested an extension of the MTLR where a deep neural networks parametrization replaces the linear parametrization and Luck et al. [106] proposed a neural network model that produces two outputs: one is the risk and one is the probability of observing an event in a given time bin. Unfortunately, the authors for most of these

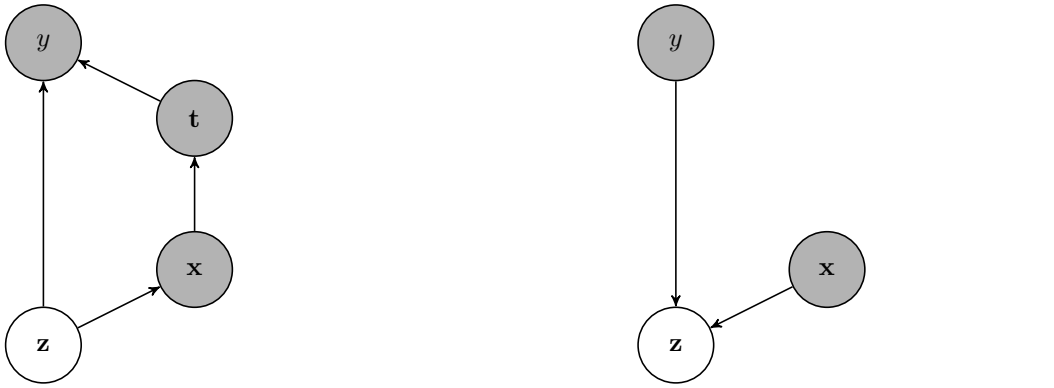
techniques [106, 156, 74] did not provide either their code or a package which causes great reproducibility problems and leads to a serious accessibility issue for practitioners. The DeepSurv architecture [80] proposed by Katzman et. al is a direct extension to the CoxPH model where the linear function of the covariance is replaced by a deep neural network. This allows the model to grasp high-order interactions between predictors therefore solving problem (3). By allowing for interaction between covariates and the treatment the proposed model provides a treatment recommendation procedure. Finally, the authors provided a Python library available on the first author's GitHub [79].

Latent-variable models

The final model is a latent-variable model based on the Variational AutoEncoder (VAE) [86, 83] architecture that we propose. Louizos et al. [104] recently suggested a latent variable model for causal inference. The latent variables allow for a more flexible observed variable distribution and intuitively model the hidden patient status. Inspired by this model and by the recommendation of Nazbal et al. [110] we implemented a latent variable model [12] that adapts the VAE architecture for the purposed of survival analysis. Our Survival Analysis Variational AutoEncoder (SAVAE) uses the latent space to represent the patient true sickness status and can produce individual patient survival function based on their respective covariates which should solve problem (1), (2) and (3). We introduce our model in the next section.

6.4 Survival Analysis Variational AutoEncoder

In this section we present our Survival Analysis Variational AutoEncoder (SAVAE). The main purpose of latent variables is to allow for more flexible and complicated distributions for the observed variables [15, 88]. The VAE we proposed is associated with the following graphical representation:



(a) Generative network. It assumes $p(x, y, t, z) = p(z)p(x|z)p(t|x)p(y|t, z)$.

(b) Inference network. Given observations x and response y we can infer the latent variable using $q(z|x, y)$.

Figure 6.1: The graphical representation of our deep-latent variable model. The response is identified by y, the treatment by t, the observed characteristics by $\mathbf{x}$ and the patient health status by $\mathbf{z}$.

The representation illustrated in figure 6.1 induces a natural factorization of the joint distribution. As said earlier, from a practical perspective, the latent variables $\mathbf{z}$ are incorporated to allow for a more flexible observed data distribution but the graphical model also leads to an intuitive description. Since

there can be two patients with the same characteristics x that received the same treatment t but have completely different response y we assume there is an unobserved or unmeasured variable that affects our collected data set. This latent variable z represents the true patient health status and directly affects the survival chances of the patient y and the covariates $\mathbf{x}$ that were gathered as proxy for the true health status. The treatment $\mathbf{t}$ is considered a special covariate as it is selected by a physician based upon the observed covariates $\mathbf{x}$. Finally, the distribution for the response y depends on the patient health status $\mathbf{z}$ and the treatment selected $\mathbf{t}$.

Neural network parametrizations along edges of the graphical representation ensure that the model includes high order of interactions between the variables which is important to us but difficult to do with the Cox PH model, described as problem (3).

6.4.1 Model distributions

In order to establish the objective function, the ELBO, we need to establish the various distributions of our model. Based upon the factorization suggested in figure 6.1, the joint distribution can be expressed as

$$p_\theta(\mathbf{z}, \mathbf{x}, \mathbf{t}, y) = p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{t}|\mathbf{x})p_\theta(y|\mathbf{t}, \mathbf{z}). \tag{6.1}$$

We decided to set the prior distribution of the latent variables to a simple Normal ball, a classic choice for VAEs

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|0, I). \tag{6.2}$$

The size of the latent space is considered a tuning parameter, using the validation set we decided upon a latent space of dimension 4. The observed predictors $\mathbf{x}$ are assumed conditionally independent given the latent variables $\mathbf{z}$

$$p_\theta(\mathbf{x}|\mathbf{z}) = \prod_{j=1}^{D_x} p_\theta(x_j|\mathbf{z}). \tag{6.3}$$

Within the data set the proxy variables for health status $\mathbf{x}$ were taking multiple forms and thus we used many different distributions: Normal distribution for continuous variables, generalized Bernoulli (categorical) distribution for categorical variables, Poisson distribution for counting variables and finally exponential distribution for time-to-event variables. In our model, $\mathbf{t}$ represents possible additional treatments; intensive chemotherapy and radiation therapy. Both of these are independently either given or not, thus a Bernoulli distribution is well suited to model these two variables

$$p(t_j|\mathbf{x}) = \text{Ber}(\hat{\pi}_j) \text{ for } j \in \{1, 2\}. \tag{6.4}$$

Finally, the distribution of the response y was set to be Weibull, a common distribution in the survival analysis literature

$$p(y|\mathbf{t}, \mathbf{z}) = \text{Weibull}(\lambda, K). \tag{6.5}$$

In order to allow for interactions between variables, for a complex relationship between the latent variables and the observed ones and for a general set up that might get expanded to more applications, we utilized a neural network parametrization. More specifically, we use neural networks along all edges of the graph in figure 6.1 to represent the relationship between the set of parent variables and the

parameters of the children distributions. Explicitly,

$$\theta = f_2(\mathbf{W}_2 f_1(\mathbf{W}_1 \mathbf{z})) \tag{6.6}$$

$$[\pi_1, \pi_2] = f_4(\mathbf{W}_4 f_3(\mathbf{W}_3 \mathbf{x})) \tag{6.7}$$

$$[\lambda, K] = f_6(\mathbf{W}_6 f_5(\mathbf{W}_5 [\mathbf{z}, \mathbf{t}])), \tag{6.8}$$

where f's are activation functions and $\mathbf{W}$'s are matrices of weights which include biases.

Finally, because the true posterior $p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{t}, y)$ is intractable analytically we proposed a variational approximation. Our proposed approximation $q_\varphi(\mathbf{z}|\mathbf{x}, y)$ is independent of $\mathbf{t}$, as illusatrated in figure 7.5b. This is because the true poseterior is also indepedent of $\mathbf{t}$ which is a direct consequence of the joint decomposition proposed $p(\mathbf{z}, \mathbf{x}, \mathbf{t}) = p(\mathbf{z})p(\mathbf{x}|z)p(t|\mathbf{x})$,

$$\begin{aligned} p(\mathbf{z}|\mathbf{x}, \mathbf{t}) &= \frac{p(\mathbf{z}, \mathbf{x}, \mathbf{t})}{p(\mathbf{x}, \mathbf{t})} \\ &= \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})p(t|\mathbf{x})}{p(\mathbf{x})p(\mathbf{t}|\mathbf{x})} \\ &= \frac{p(\mathbf{z})p(\mathbf{x}|\mathbf{z})}{p(\mathbf{x})} = p(\mathbf{z}|\mathbf{x}). \end{aligned}$$

This results in the following variational distribution

$$q_\varphi(\mathbf{z}|\mathbf{x}, y) = \mathcal{N}(\mathbf{z}|\mu, \sigma^2 I), \tag{6.9}$$

where the parametrization is established again with a neural network

$$[\mu, \sigma] = f_8(\mathbf{W}_8 f_7(\mathbf{W}_7 [\mathbf{x}, y])). \tag{6.10}$$

To summarize, $\mathbf{W}_i$ for $i \in \{1, ..., 8\}$ are matrices of parameters that require training.

6.4.2 Fitting the parameters

The parameters of the various neural networks will require training. As mentioned in section 2.2.3, the ELBO is a lower bound for the log-likelihood of the observed data and will be the objective function to maximize during training. Using the factorization proposed by figure 6.1 and explained in section 6.4.1 we have

$$\begin{aligned} \text{ELBO} &= \mathbf{E}_{q_\varphi}\left[\ln \frac{p_\theta(\mathbf{x}, t, y, \mathbf{z})}{q_\varphi(\mathbf{z}|\mathbf{x}, y)}\right] = \mathbf{E}_{q_\varphi}\left[\ln p_\theta(\mathbf{x}, t, y, \mathbf{z}) - \ln q_\varphi(\mathbf{z}|\mathbf{x}, y)\right] \\ &= \mathbf{E}_{q_\varphi}\left[\ln p_\theta(\mathbf{z}) + \ln p_\theta(\mathbf{x}|\mathbf{z}) + \ln p_\theta(t|\mathbf{x}) + \ln p_\theta(y|t, \mathbf{z}) - \ln q_\varphi(\mathbf{z}|\mathbf{x}, y)\right]. \end{aligned} \tag{6.11}$$

When fitting such model, we maximize the ELBO with respect to the matrices of weights ($\mathbf{W}$'s) of the neural network functions. This will require the use of back-propagation combined with a gradient-based optimizer.

6.4.3 Prediction and decision-making

The ultimate goal of this analysis is to provide tools to physicians that allow them to make a decision about the treatment needed for a patient. With our probabilistic approach we aim at giving physicians a wide range of information which they can utilize however they see fit. Our model produces a predicted distribution for the event-free survival time of a patient given its characteristics and the selected treatment. Having such distribution for every possible treatment gives flexibility regarding decision-making as physicians can look at various properties of the predicted distribution such as its expected value, its variance or its survival function.

Under the parametrization of equation 6.1 induced by the graphic of figure 6.1 there is no direct way to estimate $p(\mathbf{y}|t,x)$, the density for the response given the treatment and the patient characteristics. Consequently, we use an importance sampling technique

$$\begin{aligned} p(y|t,\mathbf{x}) &= \int_{\mathbf{z}} p(y|t,\mathbf{x},\mathbf{z})p(\mathbf{z}|t,\mathbf{x})d\mathbf{z} \\ &= \int_{\mathbf{z}} p_\theta(y|t,\mathbf{z})p(\mathbf{z}|t,\mathbf{x})d\mathbf{z}. \end{aligned} \tag{6.12}$$

Since we cannot sample directly from $p(\mathbf{z}|t,\mathbf{x})$ we must use a distribution of $\mathbf{z}$ from which we can easily draw samples. The prior $p_\theta(\mathbf{z})$ is easy to sample from, thus

$$\begin{aligned} p(y|t,\mathbf{x}) &= \int_{\mathbf{z}} p_\theta(y|t,\mathbf{z})p(\mathbf{z}|t,\mathbf{x})d\mathbf{z} \\ &= \int_{\mathbf{z}} p_\theta(y|t,\mathbf{z})\frac{p(\mathbf{z}|t,\mathbf{x})}{p_\theta(\mathbf{z})}p_\theta(\mathbf{z})d\mathbf{z} \\ &\approx \frac{1}{L}\sum_{l=1}^{L} r_l p_\theta(y|t,\mathbf{z}_l), \end{aligned} \tag{6.13}$$

where $r_l = p(\mathbf{z}_l|t,\mathbf{x})/p_\theta(\mathbf{z}_l)$. The above will be a mixture of Weibull of L components with weights r_l/L. One might notice that we cannot evaluate $p(\mathbf{z}_l|t,\mathbf{x})$ with our current model, but we can up to a normalization constant which leads to the following

$$\begin{aligned} p(y|t,\mathbf{x}) &= \int_{\mathbf{z}} p_\theta(y|t,\mathbf{z})\frac{p(\mathbf{z}|t,\mathbf{x})}{p_\theta(\mathbf{z})}p_\theta(\mathbf{z})d\mathbf{z} \\ &= \int_{\mathbf{z}} p_\theta(y|t,\mathbf{z})\frac{p(\mathbf{z},t,\mathbf{x})}{p_\theta(\mathbf{z})p(\mathbf{x},t)}p_\theta(\mathbf{z})d\mathbf{z} \\ &= \int_{\mathbf{z}} p_\theta(y|t,\mathbf{z})\frac{p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})p(t|\mathbf{x})}{p_\theta(\mathbf{z})p(\mathbf{x})p(t|\mathbf{x})}p_\theta(\mathbf{z})d\mathbf{z} \\ &= \int_{\mathbf{z}} p_\theta(y|t,\mathbf{z})\frac{p_\theta(\mathbf{x}|\mathbf{z})}{p(\mathbf{x})}p_\theta(\mathbf{z})d\mathbf{z} \\ &\approx \frac{1}{L}\frac{1}{p(\mathbf{x})}\sum_{l=1}^{L} p_\theta(\mathbf{x}|\mathbf{z})p_\theta(y|t,\mathbf{z}_l). \end{aligned} \tag{6.14}$$

We can also use the same sample to evaluate the normalization constant $p(\mathbf{x})$

$$p(\mathbf{x}) = \int_{\mathbf{z}} p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})d\mathbf{z} \approx \frac{1}{L}\sum_{l=1}^{L} p_\theta(\mathbf{x}|\mathbf{z}_l), \tag{6.15}$$

and both of these results combined lead to

$$p(y|t,\mathbf{x}) \approx \sum_{l=1}^{L} w_l p_\theta(y|t,\mathbf{z}_l), \tag{6.16}$$

where

$$w_l = \frac{p_\theta(\mathbf{x}|\mathbf{z}_l)}{\sum_{k=1}^{L} p_\theta(\mathbf{x}|\mathbf{z}_k)}, \tag{6.17}$$

which resembles a mixture of Weibull, where L is the number of components, and w_l is the component weight. This is the function that allows us to predict the survival distribution for a patient with characteristic $\mathbf{x}$ which received the treatment t.

One of the benefit of SAVAE is that instead of producing point estimates, say the expected survival time, it models the time-to-event with a complete distribution. We believe it offers a lot more flexibility which in turns allow for different decision-making approaches. Here, we will mention a few examples of information that can be extracted from the predicted distribution.

To begin, we could compute the expected survival time

$$\begin{aligned} \mathbf{E}[y|t,\mathbf{x}] &= \mathbf{E}\sum_{l=1}^{L} w_l p_\theta(y|t,\mathbf{z}_l) \\ &= \sum_{l=1}^{L} w_l \mathbf{E}(y|t,\mathbf{z}_l). \end{aligned} \tag{6.18}$$

Survival function at any point in time y can also be obtained quite simply

$$P(Y > y|t,\mathbf{x}) = \sum_{l=1}^{L} w_l P_\theta(Y > y|t,\mathbf{z}_l). \tag{6.19}$$

An advantage of our proposed model is that it allows for different decision-making strategy; a physician could be interested in three years survival chances, another physician might prefer to estimate four years survival chances and one might be interested in the expected survival. All of those physicians could select the appropriate treatment according to their decision-making strategy. As mentioned earlier, if we wish to give additional treatment to a patient only if it improves drastically it's survival chance then our model can estimate the increase in survival chances at a given time y by selecting treatment t_1 instead of treatment t_0 by computing

$$P(Y > y|t_1,\mathbf{x}) - P(Y > y|t_0,\mathbf{x}). \tag{6.20}$$

In short, a practitioner could decide ahead what is considered a significant improvement in survival chances, say α, and give treatment t_1 instead of the base treatment t_0 if equation 6.20 is greater than α.

6.5 Data analysis

6.5.1 Evaluation metrics

We use two different metrics to evaluate the various algorithms, both are well established and they evaluate different properties of the models. First, the concordance index [60] is a metric of accuracy for the ordering of the predicted survival time or hazard. Second, the Brier score [57] is a metric similar to the mean squared error but adapted for right-censored observations.

Concordance Index

The concordance index (c-index) was proposed by Harrell et al. [60]. It is one of the most popular performance measures for survival problems [138, 26, 79] because of the way it accounts for the censored data. It is defined as the proportion of all usable patient pairs in which the predictions and outcomes are concordant. Pairs are said to be concordant if the predicted event times have a concordant ordering with the observed event times.

Recently Steck et al. used the c-index directly as part of the optimization procedure [138], their paper also elegantly presents the c-index itself as illustrated in figure 6.2. In their article, it is defined as the fraction of all pairs of subjects whose predicted survival times are correctly ordered among all subjects that can actually be ordered. We expect a random classification algorithm to achieves a c-index of 0.5. The further from 0.5 the c-index is the more concordant pairs of predictions the model has produced. A c-index of 1 indicates perfect predicted order.

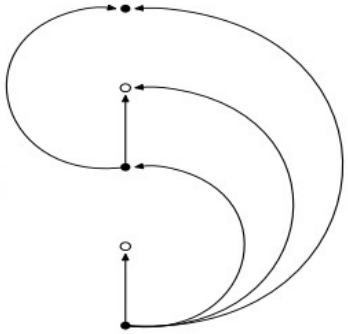


Figure 6.2: Steck et al.(2008) graphical representation of the c-index computation. Filled circle represents observed points and empty circle represents censored points. The edges in the figure represent the pairs of points for which the order of events can be established.

Figure 6.2 illustrates when we can compute the concordance for a pair of data points; this is represented by an arrow (edge). We can evaluate the order of events if both events are observed. If one of the data points is censored, then concordance can be evaluated if the censoring for the censored point happens after the event for the observed point. If the reverse happens, if both points are censored or if both events happen exactly at the same time then we cannot evaluate the concordance for that pair.

Brier Score

The Brier score established by Graf et al. [57] is a performance metric inspired by the mean squared errors (MSE). For a survival model it is reasonable to try to predict $P(T > t|X = x) = S(t|X = x)$ the survival probabilities a time t for a patient with predictors x which is represented with $\pi(t|x)$ in Graf's notation. Thus, $\hat{\pi}(t|x)$ is the predicted probability of survival at time t for a patient with characteristics x. These probabilities are considered predictions for the observed event $y = \mathbf{1}(T > t)$ in order to compute the MSE. If the data contains no censoring then the Brier Score is:

$$\mathrm{BS}(t) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{1}(T_i > t) - \hat{\pi}(t|x_i))^2 \tag{6.21}$$

Assuming we have a censoring survival distribution $G(t) = P(C > t)$ and an associated Kaplan-Meier estimated $\hat{G}(t)$. For a given fixed time t we are facing three different scenarios :

Case 1: $T_i > t$ and $\delta_i = 1$ or $\delta_i = 0$

Case 2: $T_i < t$ and $\delta_i = 1$

Case 3: $T_i < t$ and $\delta_i = 0$,

where $\delta_1 = 1$ if the event is observed and 0 if it is censored. For case 1, the event status is 1 since the patient is known to be alive at time t; the resulting contribution to the Brier score is $(1 - \hat{\pi}(t|x_i))^2$. For case 2, the event occurred before t and the event status is equal to $\mathbf{1}(T_i > t) = 0$ and thus the contribution is $(0 - \hat{\pi}(t|x_i))^2$. Finally, for case 3 the censoring occurred before t and thus the contribution to the Brier score cannot be calculated. To compensate for the loss of information due to censoring, the individual contributions have to be reweighed in a similar way as in the calculation of the Kaplan-Meier estimator leading to the following Brier Score :

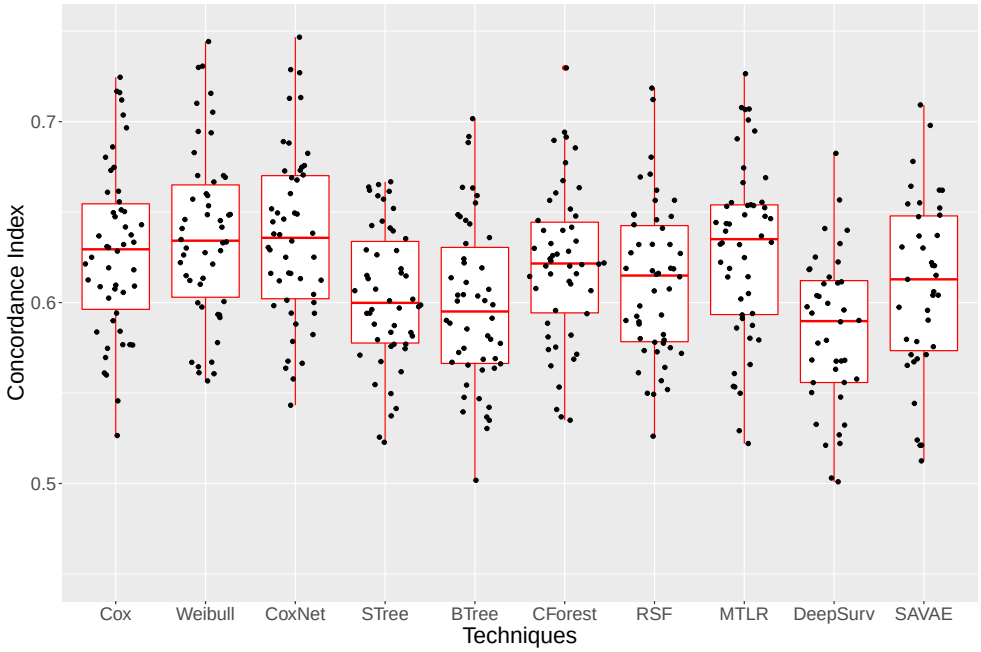
$$\mathrm{BS}^c(t) = \frac{1}{n}\sum_{i=1}^{n}\Big((0 - \hat{\pi}(t|x_i))^2\mathbf{1}(T_i < t, \delta_i = 1)(1/\hat{G}(T_i)) + (1 - \hat{\pi}(t|x_i))^2\mathbf{1}(T_i > t)(1/\hat{G}(t))\Big) \tag{6.22}$$

6.5.2 Comparative results

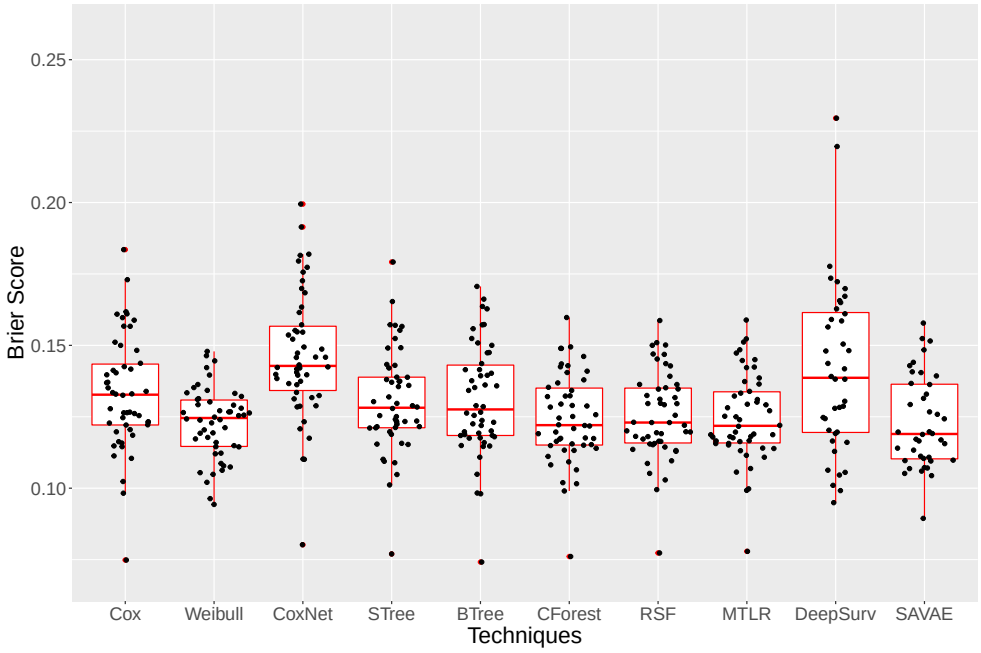
The data set introduced in section 6.2 was imported in both R [118] and Python [150]. To evaluate the algorithms we randomly divided the data set into 1500 training observations and 212 testing observations. The models were fit using the training observations and the evaluation metrics were computed on the testing observations.

As mentioned in the previous sections, the CoxPH benchmark and the conventional statistical learning models were all tested in the R language [118]. They were relatively easy to use with very little adjustment needed and clear and concise documentation. The computational speed of these algorithms was fast enough on a single CPU so that we could perform 50 trials. The newly established techniques needed a deeper understanding of the model as they contain many hyper-parameters that require calibration. They were also slower to run on a single CPU.

Figure 6.3 illustrates Sinaplots [134] with associated Boxplots of the c-index and the Brier scores for the CoxPH model and the 8 competitors. We used standard boxplots on the background since they are common and easy to understand. The sinaplots superposed on them represent the actual observed metric values and convey information about the distribution of the metrics for a given technique. As



(a) Boxplots and Sinaplots of the c-index (higher the better).



(b) Boxplots and Sinaplots of Brier Scores evaluated at 3 years (lower the better)

Figure 6.3: Results from our experiments.

mentioned earlier c-index ranges from 0.5 to 1 where a c-index of 1 indicates perfect predicted order. According to figure 6.3a, it seems no model clearly outperforms another. CoxNet is the best-performing model but the difference is not statistically significant with either the CoxPH nor the Weibull model.

Since the Brier score is a metric inspired by the mean squared error, it ranges from 0 to 1 and the lower the Brier score is the better the technique is. The SAVAE has the lowest average Brier

score of all the techniques compared and this is the main reason why we published our results in a NeurIPS workshop. Additionally, the SAVAE significantly (statistical significance) outperforms the CoxPH benchmark. However, as shows in Figure 6.3b none of the new techniques drastically outperforms CoxPH and none significantly outperform the secondary Weibull benchmark.

6.5.3 Specifics about SAVAE

Our proposed SAVAE shows potential and achieved the best Brier Score of all the models we tested. However, compared to the benchmark CoxPH model, we had to invest hours to adjust the hyper-parameters. This is also true for the other newly established model tested. Therefore, we could not recommend our model to any practitioner; CoxPH is easy to use for any survival analysis but given a new problem and a new data set our model would need to be reconstructed all over again. This is a problem that affects most of the recent models.

6.6 Takeaways and Recommendations

The previous section demonstrates that the new ML methods offers little improvement compared to the benchmark CoxPH model according to our two designated performance metrics. This initially turned out as a disappointing result as we hoped to provide improvement in prognostic prediction and eventually positively impact patients with Hodgkin Lymphoma. On the flip side, from a critical ML perspective this is an important result as we need to evaluate the abilities of ML techniques to solve real-life data problems.

Similar results on real-life data sets are observed in article presenting methodologies [46, 106, 74] where the proposed techniques provide non-significant improvements over simple models such as CoxPH. Christodoulou et al. [44] recently performed an exhaustive review of 927 articles that discuss the development of diagnostic or prognostic clinical prediction models for binary outcomes based on clinical data. The authors of the review noted the overall poor comparison methodologies and the lack of significant difference between a simple logistic regression and newly established ML techniques in many recent publications. These results are supported by Hand [59] who discussed in detail the overall strength of the simple models, given how simple they are, compared to recent newly established ML models. This raises an important question our case study highlights: is it worth using more complex models for a slight improvement?

The alternative we proposed in section 6.3 are all more complicated than CoxPH in various manners. The new techniques require deeper knowledge of the algorithm behaviours to correctly fix the many hyper-parameters. They lead to less interpretable results due to the model complexity. Finally, they require a lot more computing power. Indeed, if the CoxPH model can be fit in seconds, most of the conventional statistical learning models take minutes to fit and the newly established models take hours. Finally, most of the new techniques still suffer from accessibility issues. For instance, as an open language Python offers very little support to users and the libraries are not maintained, not standardized and come with dependency issues.

Hand [59] demonstrates the high relative performances of extremely simple methods compared to complex ones and mathematically justifies his argument. He also discusses how these slight improve-

ments over simple models might be undesirable as they might be attributed to overfitting common ML data sets which would cause reproducibility issues on new data sets. These slight improvements might also be artificial as they were achieved only because the inventors of these techniques were able to obtain through much effort the best performance from their own techniques and not their competitors' techniques. Overall if the improvements over simple techniques are marginal, perhaps they are simply not an improvement and this argument seems to be supported by both our case study and the recent review of Christodoulou et al. [44].

On the flip side, significant improvements for diagnostic tasks have been accomplished using A.I. in recent years [99, 125, 124] and thus we have to understand the difference between those experiments and ours. There is a major difference in the style of data sets that were available. In the cited articles, images (mammographic, gigapixel pathology image, MRI scans) are analysed using deep convolutional neural networks (CNN) [55]. As mentioned in chapter 2, a CNN is well suited for image analysis as its architecture itself is designed to incorporate spatial correlation and some degree of shifts and scale invariance. In comparison, conventional techniques such as logistic regression or CoxPH are not able to grasp the signal in images, which contains a large number of highly correlated predictors that individually contain close to no information but analysed together contain a lot.

In our case study, the *stratum* predictor is a binary predictor indicating if the patient had a rapid early response to the first round of treatments. Scans of the affected regions are analysed before and after the first round of treatments and this rich information is transformed into a simple binary variable. As new tools are established to extract information from ever growing, both in size and complexity, data sets, practitioners have to rethink how they gather data and transform it to make sure that no information is lost in order to utilizes these new tools. Extracting and keeping as much information as possible and selecting models and tools that are designed to analyse a specific style of data were some of the factors in the success of CNNs in medical images analysis publications.

6.7 Conclusion

In this chapter, we identify a series of statistical and ML techniques that should alleviate some of the flaws of the well-known CoxPH model. We also establish our own model made especially to alleviate all of the identified flaws. Even though our proposed SAVAE model offers some improvement according to the Brier score most of the tested models provide little to no improvement with respect to the designated metrics. We tested techniques that should have increased our prediction abilities, instead we are forced to admit that the CoxPH performs really well even when compared to modern models. These results are supported by other articles with similar findings.

It is now clear that we cannot directly apply new algorithms on existing data set to improve our current situation. Modern models are usually hand-tailored to new problems and performances are directly tied to our abilities to craft models to answer specific questions. This is a problem if we want to establish models that are robust and generalize well to new applications and data structures. However, judging by the success of CNNs in multiple prognostic tasks, modern ML techniques can provide significant improvements when used in the right situation.

Appendix

Variable	Type	Description
agedxyrs	Continuous	Age of the patient at the start of the treatment
gender	Binary	Biological gender
stage	Categorical	Cancer stage ranging from 1 to 4
b_symptoms	Binary	Presence of B symptoms
bulk_disease	Binary	Presence of Bulk disease
extralymphatic_disease	Binary	Presence of Extralymphatic disease
fever	Binary	Presence of recurrent fever
night_sweats	Binary	Presence of night sweats
weight_loss	Binary	Presence of significant weight loss (> 10%)
nodal_aggregate	Binary	Presence of a nodal aggregate
mediastinal_mass	Binary	Presence of a mediastinal mass
esron	Continuous	Erthroctye sedimentation rate (mm/hr)
istnon	Continuous	Number of involved nodal sites
histology	Categorical	Histology (LP,LD,NS,MC, unknown)
albon	Continuous	Albumin (g/dL)
hgbon	Continuous	Hemoglobin(g/dL)
amend	Binary	
stratum	Binary	Rapid early response to first treatment
morpho_icdo	Categorical	ICD-O Morphology codes
RT	Binary	Treatment variable: Radiotherapy
DECA	Binary	Treatment variable: Intensive Chemotherapy

Table 6.1: Predictor variables and description

Chapter 7

HWD+ data set: a new computer vision data set

In the chapter, we present a new hand-written digit data set. It contains high-resolution images of hand-written digits, a writer identification and various writer characteristics. The data set is publicly available and is designed to create new research opportunities. We also perform a thorough analysis of this new data set. We begin with simple supervised tasks. We assess the predictability of the writer characteristics gathered, the effect of using some of those characteristics as predictors in classification tasks and the effect of higher resolution images on classification accuracy. We then explore semi-supervised applications; we can leverage the high quantity of hand-written digits data sets already existing online to improve the accuracy of various classifications task. Finally, we demonstrate the generative perspective offered by this new data set.

In this chapter we (1) introduce a new data set we collected; the data set provides new research opportunities and we (2) provide a thorough analysis of the data set that establishes benchmarks and showcases some of the new opportunities made possible with this new data set. Finally, we (3) provide preliminary results of *controllable content generation*, one of our long-term ongoing research projects. The main contributions of this chapter were introduced first in our article *Analysis of a high-resolution hand-written digit data set with writer characteristics*[10] currently under review.

7.1 Introduction

Modern computer vision algorithms have become impressively good at identifying the content of a complex image. A scanned hand-written document is an example of a complex image for which many algorithms were developed. In this case, the task assigned to the algorithm is to identify letters, digits and later words and sentences. In hand-written document analysis, the MNIST data set introduced by LeCun & al. [95] quickly became a benchmark for hand-written digits recognition and is now a rite of passage for computer vision algorithms. Usually, MNIST is used for a simple task, try to identify the digit in new hand-written digit images given a training set of labelled hand-written digit images.

In this project, we explore the potential of modern computer vision algorithm for a wider range of inference tasks on hand-written digits. For instance, we will tackle the writer identification problem

which is a common problem in criminology or historical research. Broadly speaking, if more labels were attached to an image, could we successfully extract other useful information out of those images? Our goal is to rely on modern computer vision algorithms to replace feature engineering (handcrafted features). Based on the recent results obtained by Adak et al. [1] it appears that auto-derived features outperform feature engineering and this what we are hoping to exploit here.

We tackle the well-established task of writer identification, but also statistical inference of writer characteristics. We also discuss new research opportunities created with this data set. Typically, computer vision algorithms are build to identify the content of the images but the tasks we tackle here are slightly more complex has we hope to predict writer characteristics that should affect only subtle details of the image. Our contribution is two-fold; first, we introduce and distribute a new data set that we collected, HWD+, containing hand-written digit images in high-resolution and various writer characteristics. This data set can be utilized as a standalone data set and also in conjuncture with MNIST for semi-supervised learning projects. Second, we perform a first analysis of the data set under both the supervised and semi-supervised paradigm. We also showcase how to use this data set to experiment with controlled image generation.

The remaining of this chapter is organized as follows: we discuss the related publications in Section 2. Section 3 introduces the new data set we collected. Following this, we introduce the algorithms used for our first analysis in Section 4. Section 5 contains the results of our analysis.

7.2 Related work

In the contributed data set, we collected various characteristics about our writers and also assigned a writer ID to each writer. Consequently, one natural problem to tackle is writer identification. This problem has been extensively studied in the past and is still a relevant problem in forensics. A recent publication from Adak et al. [1] attempts to solve a writer identification problem and compares the performances of models that rely on hand-crafted feature against models with auto-derived features. Slightly before that, Xiong et al. [153] produced one of the most recent surveys comparing various modern writer identification algorithms. A result shared across both articles [153, 1] and highlighted in a comprehensive review [122] is that auto-derived feature models perform better than feature engineering and thus we rely on auto-derived feature models in this analysis.

One of the tasks we've established for this research project was to assess the abilities of modern computer vision algorithms to infer some of the writer's characteristics. Most literature that discusses writer characteristics addresses graphology; the analysis of hand-writing patterns in order to identify psychological traits of writers. However, serious studies on graphology demonstrate that it is more a pseudo-science than anything else [87]. As a result, we focus this works on measurable characteristics such as age, gender or native language. We are interested in determining the feasibility of predicting such characteristics based on handwritten digits.

When considering the identification of the digits themselves, the MNIST data set inspired a gigantic amount of publications. The first article to discuss this data set [95] was published in 1998 and introduced the data set and compared the prediction accuracy of multiple classification methods. The two best

performing algorithms were a committee of deep convolutional neural networks (CNN) and a support vector machine (SVM) with test error rates as low as 0.7% and 0.8% respectively. This article really set the tone for future computer vision publications by establishing the sheer dominance both in terms of accuracy and memory requirements of CNNs. It was a pivotal point into explaining and empirically proving the benefits of automated feature extraction. It has also established the MNIST data as an important benchmark data set.

Since then the best results obtained from a SVM algorithm was obtained in 2002 [38] with a 0.56% error rate. Simple techniques that require no training, such as KNN, have achieved higher accuracy (0.54% error rate) by allowing the algorithm to search into a set of distorted images [81]. The lowest error rate (0.35%) achieved by a single NN was reported in 2010 [31]. Finally, in 2012 a committee of 35 CNNs achieved a 0.23% test error rate [32]. A problem with MNIST is that current algorithms achieve a classification accuracy that is so high that it leaves room only for marginal improvements. The true usefulness of these improvements is hard to evaluate [59] as it might be caused to details that are specific to the MNIST data set and thus aren't real improvement applicable to new problems. In other words, it is possible that MNIST has been overused and that some new models are *overfitting* this data set.

Finally, let us address related data sets. As we already mentioned, our data set is quite similar to the MNSIT data set [94]. Other digit image data sets also became quite popular such as the SVHN data set [111] which contains images of house numbers in Google Street View images. The only label included in those data set is the digit itself and supervised tasks are directed at digit classification. Our data set, HWD+, offers more opportunities since it contains various characteristics and writer identification. Furthermore, our data set also contains high-resolution images.

For writer identification, there exist multiple text-written data set. For instance, the CEDAR (Center of Excellence for Document Analysis and Recognition) [25, **?**] developed multiple data sets containing either only letters, continuous text or signatures.

There exist various massive multi-labels data sets online such as CelebA [101], DeepFashion [100] and DeepFashion2 [52] that contains images in high resolution and multiple labels. However these data sets can be overwhelmingly large and require anyone who wishes to use them to have access to multiple GPUs in order to experiment with them. Our new data set, the HWD+ data set, is much more approachable. It offers multiple labels and high resolution images but also a 28 by 28 pixel alternatives for those who wishes to simply plug in this new data set in a code already set up for MNIST or SVHN.

7.3 Data set

We named our Hand-Written Digits data set HWD+. The plus sign stands for the additional writer characteristics collected. To collect a valuable data set, we followed some recommendations included in a recent article published by Rehman et al. [123]. The same authors noted in another article [122] how few existing data sets have a large enough data size to utilize modern computer vision architecture for writer identification; our data set is a contribution in that aspect.

The HWD+ data set contains 13,580 images from 97 different writers. Images were collected in a high resolution of 500 by 500 pixels in a shades-of-grey format. We also collected various information

about the writers. We believe our data set has a weak signal for some variables and thus leave plenty of room for improvement in contrast to the popular MNIST data set where almost all algorithms achieve a good performance and where top-of-the-line algorithms achieve such a high accuracy that it becomes difficult to distinguish their performances.

We believe that the large resolution and the set of writer characteristics collected will lead to new questions and findings. It is a unique data set that could be used in multiple fashions; this is why this section carefully explains how to the data was gathered and processed into the data set now publicly available online [8].

7.3.1 Data gathering

Our data gathering efforts were drastically affected by the 2020 COVID-19 pandemic. We hoped to sample a large number of volunteers, had already bought the necessary material and planned our data gathering procedure. Unfortunately, the social distancing efforts forced us to settle on a smaller size data set with a reduced number of writers that were not randomly sampled. For this reason, it would not be responsible to use this data set for inference or to establish causality. Thankfully, we can still establish the predictability of various variables, compare computer vision models and much more.

Outside of uncontrollable events we made sure to gather a data set in a standardized manner that we believe contains interesting information. Every writer was given 2 pages containing a one inch square grid of 10 rows by 7 columns. Writers were asked to fill these pages with digits, 2 rows per digits for a total of 14 replications per digits as seen in Figure 7.1. Every writer was given a new Sharpie pen.

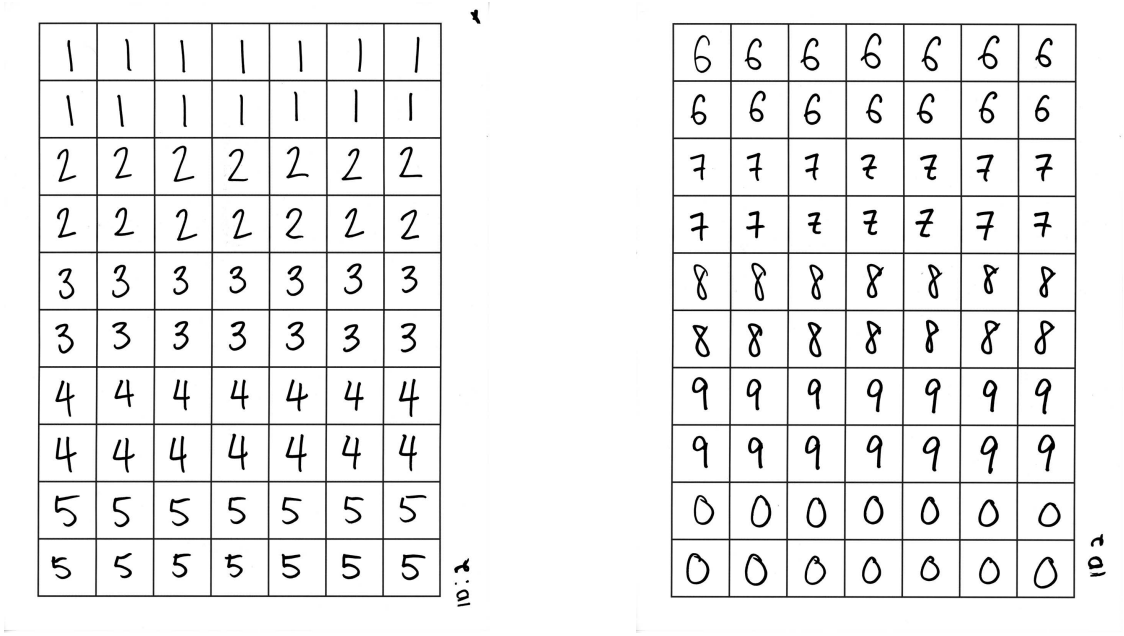


Figure 7.1: Example of the collected images for a single writer.

Following this, the pages of handwritten digits were attached to their user identification (ID). We also collected the following writers characteristics : (1) age, (2) biological gender, (3) height, (4) language

learned in elementary school, (5) handedness of the writer, (6) education level and (7) main medium used to write. Characteristics (1), (2), (3) and (5) are self-explanatory. For (4) we were interested to find out if different educational system led to different digit writing styles. We initially assumed that there could be a noticeable difference between writers who were taught with the Roman alphabet and those who were taught a Chinese or an Arabic alphabet. The educational level (6) was encoded as a four-level categorical variables were the first level represents high school, the second level means the writer completed an undergraduate program, the third level is assigned to writers who completed a master's degree or a Ph.D and we finally added a fourth level for young kids who did not complete high school yet. Finally, for the most commonly used writing medium (7), writers were asked to choose between handwriting, keyboard or other where the latest category was commonly cellphone or electronic pen.

As previously mentioned, the COVID-19 pandemic drastically slowed down our data collecting effort and at the moment of submitting this article we are still actively collecting more data. We plan to update our database in the coming months in order to further increase data size. The one currently available contains 97 different writers for a total of 13,580 images.

7.3.2 Data processing

All of the pages collected were scanned using the same machine with the same settings: shades-of-gray and 600 pixels per inches. These pages were then processed through a script that would take off the edges of the pages and divide the grid into 600 by 600 pixels squares. We trimmed of 50 pixels off the four sides of every image to trim off the actual grid and the result is a collection of 500 by 500 pixels images.

Those images were imported in Python where they were attached to their writer ID, the seven characteristics previously discussed and the digit label. These images are stored as shades of grey images, thus they are composed of a single channel taking values between 0 and 255. When images are scanned, some of the white parts of the images lose some of their purity and thus we have set to 255 every pixel that had a value above 200. The digits were not centred, not scaled and not rotated. These 500 by 500 pixels images form the completed data set available.

For simplicity we also produced two other data sets with different images size. One data set contains 100 by 100 pixels images. This still is a rather high resolution but it is much faster to run computer vision algorithms on these images than on their 500 by 500 counterparts. We also produced a data set of size 28 by 28 as it is the size of images in the MNIST data set. This allows researchers to use already existing code set up for MNIST and simply swap data sets. The 28 by 28 data set could also be used in conjecture with the MNIST data set for semi-supervised projects. The fact that it is similar to MNIST but very different at the same time should allow us to understand the problems related to the massive use of MNIST in the recent years. Image compression was done using the open CV [17] Python library.

We have done very few pre-processing compared to other popular data sets by choice. To begin, we believe that size and skwedness are genuine writing characteristics that might contain valuable information about the writer and we did not want to discard that information. Thus, we decided to release the data sets detailed above with as little pre-processing as possible.

Figure 7.2 contains a sample of what the images in the data set look like.

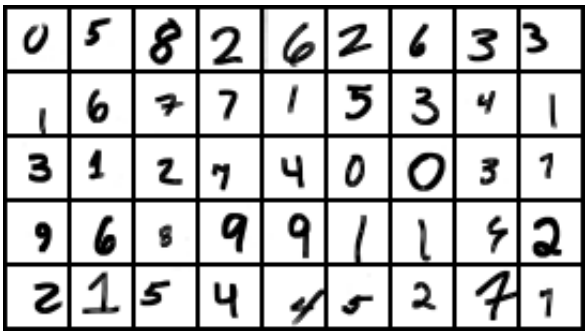


Figure 7.2: Sample of 45 images.

7.4 Computer Vision Algorithms

Two models are central for our experiments. We briefly introduce them in this section and explain why we use them. Our analysis is divided in two parts; a supervised learning analysis and an unsupervised learning analysis.

7.4.1 Convolutional Neural Networks for supervised learning

Multilayer neural networks (NN) have been extremely popular in recent years as a universal function estimator that can be fit using gradient-based approaches. They can be used as a model themselves or as part of other models, for instance they are used in VAEs as explained in Section 7.4.2. In this project, we will use NNs as prediction function where the inputs are the pixels of an image and the output is either the label, the writer ID or any other variable we are trying to predict. More specifically we will use CNNs as introduced in Section 2.1.5.

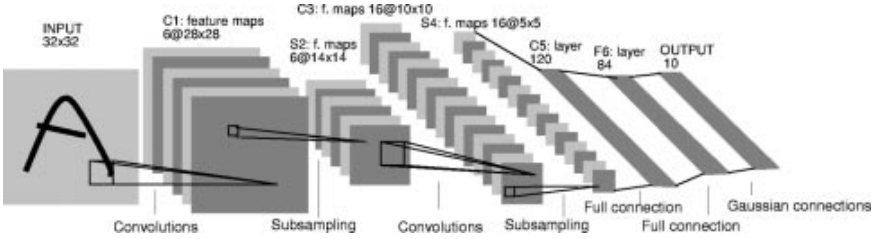


Figure 7.3: A figure provided by LeCun [95] which illustrates LeNet-5, a CNN architecture.

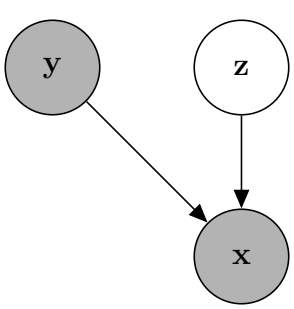
LeNet-5 illustrated in Figure 7.3 was introduced by LeCun in the same paper that introduces MNIST [95]. It contains a succession of convolution layers, pooling stages and conclude with fully connected layer before the 10-level output. By changing the size of the output layer we adapt LeNet-5 to multiple classification tasks. We use architectures similar to LeNet-5 as classifiers for our experiments.

7.4.2 Variational AutoEncoders for semi-supervised learning

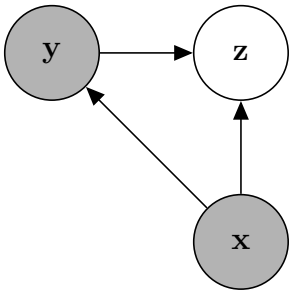
In our experiments we will be working with slight variations of the VAE introduced in Section 2.2.3 where we also include a set of selected labels $\mathbf{y}$ such as the digit or the writer ID or the digit. These models were established for semi-supervised problems. Briefly, the idea is to make use of an unlabelled data set S_u in order to improve the prediction accuracy we would get by training only with the labelled

data set S_l. We will also examine the generative abilities of such model; we are curious to find out how much more control over the generative process we gain by adding labels y into the model.

We coded and experimented with two different models, first the M2 model proposed by Kingma [85, 83].



(a) Generative network. It assumes $p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{y}) = p_\theta(\mathbf{z})p_\theta(\mathbf{y})p_\theta(\mathbf{x}|z, y)$.



(b) Inference network. Given observations x and label y we can infer the latent variable using $q_\varphi(\mathbf{z}|x, y)$. When y is missing we can infer is using $q_\varphi(\mathbf{y}|x)$.

Figure 7.4: Graphical representation of the two networks that makes up the M2 model.

To obtain an objective function for semi-supervised learning we have to consider both label and unlabelled data separately. In the first case, we have the label and the resulting ELBO is

$$\ln p_\theta(\mathbf{x}, \mathbf{y}) \geq \mathbf{E}_{q(\mathbf{z}|\mathbf{x},\mathbf{y})} \left[\ln p_\theta(\mathbf{z}) + \ln p_\theta(\mathbf{y}) + \ln p_\theta(\mathbf{x}|z, y) - \ln q_\varphi(\mathbf{z}|x, y)\right] = \mathcal{L}(x, y), \tag{7.1}$$

and for unlabelled data, the ELBO is

$$\begin{aligned} \ln p_\theta(\mathbf{x}) &\geq \mathbf{E}_{q(\mathbf{z},\mathbf{y}|x)} \left[\ln p_\theta(\mathbf{z}) + \ln p_\theta(\mathbf{y}) + \ln p_\theta(\mathbf{x}|z, y) - \ln q_\varphi(\mathbf{z}, \mathbf{y}|x)\right] \\ &= \sum_y \left[q_\varphi(\mathbf{y}|x)(\mathcal{L}(x, y))\right] + \mathcal{H}(q_\varphi(\mathbf{y}|x)) = \mathcal{U}(x), \end{aligned} \tag{7.2}$$

where $\mathcal{H}$ is the entropy of the distribution.

The bound on the entire data set is

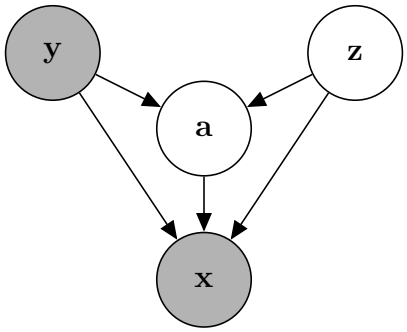
$$\mathcal{J} = \sum_{S_l} \mathcal{L}(x, y) + \sum_{S_u} \mathcal{U}(x). \tag{7.3}$$

To complete our brief introduction of the M2 model, one might notice that the encoding function used as classifiers $q_\varphi(\mathbf{y}|x)$ only appears in $\mathcal{U}(x)$ and thus is never actually trained on labelled data. To rectify this situation, Kingma proposed to add a term to $\mathcal{J}$ resulting in the following objective function

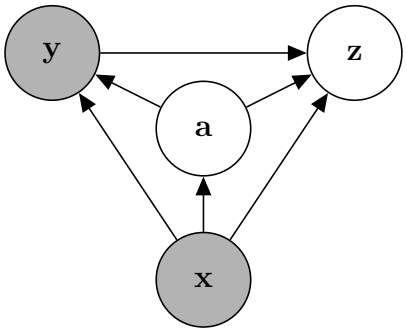
$$\mathcal{J}^\alpha = \mathcal{J} + \alpha \mathbf{E}_{S_l} \left[-\ln q_\varphi(\mathbf{y}|x)\right], \tag{7.4}$$

where α is an hyper-parameter that controls the relative weight between generative and discriminative learning. The bigger α is the closer we are to obtain the same classifier obtained using strictly the labelled data; in a way the whole VAE machinery can be perceived as regularization that prevents overfitting the training labelled point. More details about the M2 model can be found in various publications [85, 83, 121].

Finally, we have implemented the SDGM proposed by Maaløe et al. [107, 108, 121]. The model relies on auxiliary variables [2] to improve the expressive power of both the inference and generative model.



(a) Generative network. It assumes $p_\theta(\mathbf{x}, \mathbf{z}, \mathbf{y}, \mathbf{a}) = p_\theta(\mathbf{z})p_\theta(\mathbf{y})p_\theta(\mathbf{a}|z, y)p_\theta(\mathbf{x}|z, y, a)$.



(b) Inference network. We can infer y using $q_\varphi(\mathbf{y}|a, x)$ and the latent representation z using $q_\varphi(z|a, x, y)$.

Figure 7.5: Graphical representation of the two networks that makes up the SDGM model.

Figure 7.5 is the graphical representation of the SDGM. Similarly the objective function has a component for labelled observations, a component for unlabelled observations and an extra term to ensure that $q_\varphi(\mathbf{y}|a, x)$ is trained with labelled observations. More details about the SDGM can be found in various publications [107, 108, 121].

7.5 Experiments

In this section we tackle both supervised and semi-supervised learning problems. All of our experiments were performed using Python [150] and the Pytorch library [115]. After experimenting with multiple optimizers, we settled on Adam [84].

There are two main purposes for these experiments. First, we want to explore our data set, get to know its structure better, detect some of the patterns there might exist and establish the first benchmarks for some of the classification problems. Second, we want to showcase some of the new problems that can be tackled with this new data set.

7.5.1 Supervised learning

In this section, we explore our data and establish the first benchmarks attainable for various classification tasks. We approach multiple simple classification problems using four models that were previously successful; we implemented Le-Net5 [95], a deep fully-connected NN based on the work of Ciresan et al. [31], a committee of 25 CNN [32] and finally a committee of 25 deep NN. Le-Net5 [95] was selected as our default CNN; it is introduced in the same paper that introduced the MNIST data set. We included a deep NN based on the work of Ciresan et al. [31] who demonstrated that a very deep and large NN performs as well as a CNN for digit prediction. We included a committee of CNN since ensemble models have the best classification accuracy on the MNIST data set. Finally, we included a committee of deep NN for comparative purposes.

Second we address some possible new problems we can approach with this new data set. We assess

how higher resolution affects classification performances and how using writer characteristics as predictors affects the prediction accuracy. We do not address multi-label classification problems in this article, but this it is another problem that can be tackled with this data set that could not be tackled with MNIST.

The single Le-Net5 CNN and the deep NN are fit 50 times where each time we randomized which images are in the training set and the testing set. We fit both the ensemble models 15 times with once again randomized training and test set for each trial.

Image classification

As already mentioned we wish to establish the first benchmark but also the existence of some signal; thus we often time compare our results with the *naive classifier*, which we define here as a classifier that always votes on the majority class. Readers are invited to take a look at the descriptive statistic table in the appendix to get a rough idea of the performance of such naive classifier in this analysis.

For some of these experiments we divide our data set into a training set and a testing set in a way that both sets contain every writer; the training set contains 10 images of every digit of every writer and the test set contains 4 images of every digit of every writer. We named this process *partitioning by digits*. This partitioning will be used when predicting the digit and the ID. To better assess the actual predictability of the writer characteristics we created another way to partition training data from test data; this time we split training and test sets by participants, randomly assigning 70% of the writers to be in the training set and 30% in the test set. This way the writers in the test set have never been observed during training. We refer to this as *partitioning by individuals*.

	LeNet-5		Comm. LeNet-5		Deep NN		Comm. Deep NN	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Digit	0.9399	0.0143	0.9762	0.0013	0.9192	0.0160	0.9340	0.0029
ID	0.3473	0.0136	0.6195	0.0063	0.4268	0.0077	0.5012	0.0049
Gender	0.5367	0.0183	0.5483	0.0372	0.5394	0.0208	0.5309	0.0219
Language	0.6792	0.0322	0.7621	0.0626	0.6752	0.0604	0.7149	0.0408
Hand	0.7940	0.0285	0.8304	0.0499	0.7973	0.0275	0.8232	0.0377
Education Level	0.4117	0.0222	0.4726	0.0343	0.4147	0.0393	0.4253	0.0405
Writing Medium	0.4585	0.0225	0.4782	0.0372	0.4668	0.0189	0.4714	0.0249

Table 7.1: Mean and standard deviation of the prediction accuracy for simple classification tasks

In the table 7.1 we see that this data set has a very high signal with respect to the digit. HWD+ is of much smaller size and less processed than MNIST but nonetheless a committee of CNNs reaches close to 98% accuracy on average. Strictly for digits classification our data set is a good alternative to MNIST and should allow researchers to easily assess the impact of image processing on classifier performances.

Next, let us look at writer identification. The performance of the committee is quite impressive as observed in table 7.1, accurately predicting the writer 62% of the time, given we have a pool of 97 writers this is way above the performances of the naive classifier.

Let us now discuss the prediction of the various characteristics collected. As we previously discussed, we used a different data set partitioning for the writer characteristics. The reason is quite simple; the

CNN techniques are so good at predicting distinct style related to IDs, as shown by the high performance of the committee, that the model could map images to IDs and then IDs to characteristics. This is not exactly identifying writing patterns that are specific to some of the writer characteristics. Thus, we implemented *partitioning by individuals* for writer characteristics to make sure the algorithm actually tries to learn effects of the characteristics on the writing styles that are shared among writers.

Most of the results are worst or even with the naive classifier who simply selects the majority class. However, we noticed in table 7.1 that the improvement when using a committee of CNNs over a single CNN is statistically significant when predicting every characteristics except gender and writing medium; thus there might some signal for native language, handedness and education level. The improvement in table 7.1 is specifically important when predicting the writer ID; almost doubling the predictive performance over the simple LeNet-5. We believe the variables for which we observe a significant increase in prediction accuracy when using a committee are predictable and this improvement is a consequence of the existence of a signal. There might be some ways to further improve the prediction accuracy in order to surpass the naive classifier for those classification tasks.

We achieved one of our main goals to create a data set with variables with various predictability: we can achieve high accuracy when predicting the digit, the ID seems to lead to widely different prediction performances but is predictable, some characteristics, such as native language, handedness and education level, are weakly related with the images and finally the gender and usual writing medium seem to be too noisy to be predicted using only digit images.

We also noticed the relatively good performances of the deep NN which supports the results of Ciseran et al. [31]. Additionally, we included committees of deep NNs to better understand the difference between LeNet-5 and the deep NN. We know that unstable algorithms tend to benefit more from aggregating [19] and here we see that LeNet-5 benefits more from the aggregation than the deep NN. This would suggest that fully connected deep NNs are more stable than CNN classifiers. In other words, with slightly different data sets, CNN classifiers are more different than deep NNs which stay relatively the same.

High resolution images classification

In the next two sections, we experiment with tasks that are specific to our new data set. In this section we assess the effect of higher resolution images on the classification performances of CNNs. Being able to provide users with images as high resolution as 500 by 500 pixels is something offered by very few data sets that often contain very small images. However, in this section what we call high-resolution images are 100 by 100 pixels images.

We observe a statistically significant increase in prediction accuracy for the high-resolution image over the low-resolution one when predicting the digit, the writer ID, the first language, the writer handedness and the education level of the writer in table 7.2. These are the exact same variables for which the committee also improved on the benchmark LeNet-5 in table 7.1. Even though the predictive performance of LeNet-5 is equivalent to the naive classifier for some of those variables, those improvements when using a committee or higher resolution images lead us to believe that those variables are predictable in some way. In other words, there must exists some signal between the images and those variables.

	LeNet-5 (28x28)		Comm. (28x28)		LeNet-5 (100x100)	
	Mean	Std	Mean	Std	Mean	Std
Digit	0.9399	0.0143	0.9762	0.0013	0.9683	0.0044
ID	0.3473	0.0136	0.6195	0.0063	0.3675	0.0224
Gender	0.5367	0.0183	0.5483	0.0372	0.5354	0.0410
Language	0.6792	0.0322	0.7621	0.0626	0.7284	0.0441
Hand	0.7940	0.0285	0.8304	0.0499	0.8129	0.0355
Education Level	0.4117	0.0222	0.4726	0.0343	0.4466	0.0368
Writing Medium	0.4585	0.0225	0.4782	0.0372	0.4612	0.0234

Table 7.2: Mean and standard deviation of the prediction accuracy for simple classification tasks on low-resolution images (single LeNet-5 and committee) compared to high-resolution images (single LeNet-5).

The results here are intuitive: the classifier benefits from higher resolution images since they are richer in information. However, it should not be surprising that it also increased the computational cost. For instance, we could not fit committees of LeNet-5 classifiers on the high-resolution images on a single GPU (GeForce RTX 2070 Super 8Gb Ram) due to lack of memory. We can get around those problems by sending our tasks to servers online but we think it is important to make sure the algorithms we develop can run on a single computer as it makes the algorithms available to a broader audience. Additionally, as data sets get larger and larger we have to address the scalability of such algorithms. This data set offers the opportunity to analyse such scalability on a simple digit prediction task.

When we compare the gains made from richer information to the gains made from *better* algorithms we notice something very interesting. Training a single CNN on the high-resolution images is 25 times slower than training a single CNN on the low-resolutions images. Consequently, training a committee of 25 CNNs on the low-resolution images takes a similar amount of time than training a single CNN on high-resolution images. We see in table 7.2 that the ensemble of classifiers trained on the low-resolution data set performs better than the single LeNet-5 trained on a richer data set. Of course we expect a committee of LeNet-5 trained on the richer data set to have higher performances than the alternatives discussed but this is not the point we are trying to get across. Our results reveal that the predictive improvement provided by using an ensemble technique is higher than the improvement provided by getting a data set with twelve times as many pixels for a fixed run-time.

Image classification with predictors

In this section we include some of the collected information as predictors to see how it changes the performances of the Le-Net5 classifier, once again something new that our data set enables. Moreover, we are interested in understanding the potential contribution of additional information in images classification. For instance, we believe it would be a contribution to forensics if we establish that providing the digit (or the word) to the algorithm increases the accuracy when predicting the writer.

We experiment with two simple tasks: in the first experiment we try to classify images according to their digit and we incorporate the writer ID as an additional predictor. Next, we do the opposite, we classify images according to the writer ID while including the digit as an additional predictor. To do so, we include a one-hot encoding vector for writer ID or the digit in the first fully connected layer of LeNet-5. In other words, the additional predictors are incorporated immediately after the convolution layers; the one-hot encoding vector of predictors is concatenated with the vector C5 of Figure 7.3. For

this experiment, we *partitioned by digits* the data set.

	Images (LeNet-5)		Images + (LeNet-5)		Images (Com.)		Images + (Com.)	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Digit	0.9399	0.0143	0.9551	0.0080	0.9762	0.0013	0.9812	0.0020
ID	0.3473	0.0136	0.3575	0.0192	0.6195	0.0063	0.6003	0.0042

Table 7.3: Mean and standard deviation of the prediction accuracy for simple classification tasks when using only the image as predictors (Images) or the image and an additional predictor (Images +)

Including the writer ID as additional information significantly increases the accuracy when predicting the digit. However, including the digit when predicting the ID actually decreased the prediction accuracy of the committee.

These results warrant further investigation. For instance, there exist multiple way to integrate additional information in a CNN classifier and this data set offers an opportunity to explore those.

7.5.2 Semi-supervised learning

In this section, we tackle two semi-supervised tasks using the two semi-supervised learning models introduced in Section 7.4.2: the M2 model presented by Kingma and Wellington [85] and the SGDM model established by Maaløe et al. [107, 108, 121]. Our first problem is to perform a semi-supervised analysis where we use the MNIST data set as unlabelled observations.

The second task we focus on is image generation. We will briefly discuss and demonstrate the generative abilities of the SGDM model using our data set. The multiple labels allows us to turn multiple *control knobs* which imbue the generative process with much more control, consequently this is a contribution towards what we call *controllable content generation*.

Semi-Supervised classification

We use the M2 model described in Section 7.4.2 to predict the Digit and the ID in our images while increasing our data set size with some unlabelled images, the MNIST data set. In our implementation of the M2 model $q_\varphi(\mathbf{y}|x)$ is parametrized by a LeNet-5 CNN. We assess the improvement produced when including new unlabelled data compared to the results previously obtained in Section 7.5.1 when using a single LeNet5.

This gives us a great perspective on semi-supervised classification. It is said that it is possible to leverage unlabelled points from other data sets to improve the accuracy over the simple classifier and we have argued it is due to some regularization. However fitting the compression and decompression machinery does increase the run time needed to fit such semi-supervised model.

Table 7.4 shows a significant increase in accuracy when using the semi-supervised model. These results are surprising for us given how standardised the MNIST data set is; it is widely different from our data set with much less difference between writers. As we previously discussed the second term of the objective function presented in equation 7.4 trains the classifier on labelled data and is precisely

	LeNet-5		M2	
	Mean	Std	Mean	Std
Digit	0.9399	0.0143	0.9542	0.0060
ID	0.3473	0.0136	0.4174	0.0099

Table 7.4: Mean and standard deviation of the prediction accuracy of the semi-supervised M2 model trained on the HWD+ and MNIST data set compared to LeNet-5 trained on HWD+.

what we trained in previous sections. Further investigation on how the first term serves as regularization should lead to interesting results. We will also investigate further in a subsequent research project the idea of forming committees of classifiers fit under the semi-supervised paradigm.

Generative perspectives

In this section we showcase the opportunity our data set offers for controllable (conditional) image generation. We fit the SDGM model described in Section 7.4.2 with both the ID and the digit as labels $\mathbf{y}$. Since the model is fitted for generative purpose, we use all of our data points, which are labelled, and the classifier $q_\varphi(\mathbf{y}|x)$ is completely irrelevant here. What we truly want, is to train $p_\theta(\mathbf{x}|z, a, y)$ to generate images that are good looking and that respect the conditions imposed by y. In other word, the images have to be of the right digit with the right style. Other details of the images are randomized through $\mathbf{z}$ and $\mathbf{a}$.

To showcase our results we have produced the figures below. We picked four different IDs with drastically different styles to better illustrate that the algorithm was able to grasp some writing style details. In the figures below, the first four columns are a sample of four real images and the six following columns are generated images. We have selected the digits one, two, four, seven and nine has they exhibit large differences in style from one writer to another.

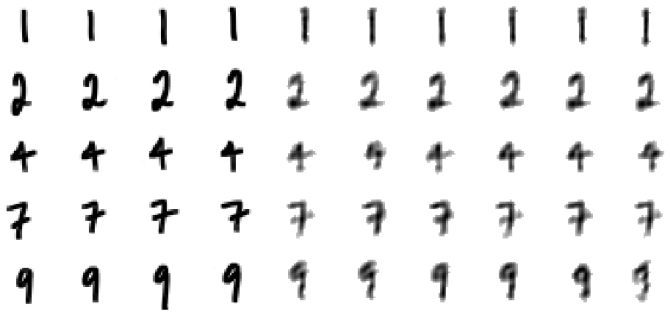
The generator seemed to have learned very well the effect of the digit input. We see that the generated digits are distinguishable and appropriate. This was to be expected based on previous experiments [85, 83].

Additionally, the SDGM also learned the writing styles of the various writers as observed in Figure 7.6 . We observe that the size of generate images respect the size of the true images as well as multiple details such as serifs and angles. For instance, the images of *ones* generated by the SDGM model has serif for ID #12 and ID #70 and not the other two. Similarly the *fours* are open for ID #29 and #70 but closed for ID #12 and #14. Moreover, *sevens* take all kind of shape, sometimes the tail of the digit *nine* is curved and so fort. Overall we are pleased with the results. We already knew it was possible to generate images of a specified digit but the writer ID is something more subtle and those images prove that the VAE model is able to grasp and mimic what makes writing styles different.

However, the generated images are blurry but this is a well-known problem for VAE generated images [136, 68, 42, 41] and a problem we are not trying to fix in this chapter. It is also a problem we mentioned previously in Chapter 5. As previously discussed we hope to fix those issues in future work.

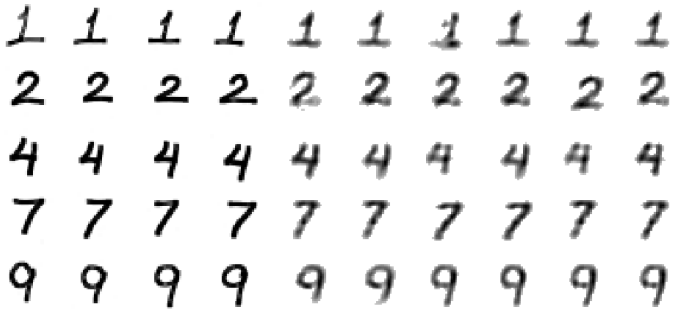
These results are preliminary but they highlight the capacity of some well-developed generative models to grasp subtle writing styles and the opportunity that our data provide to experiment with such generative models.

(a) Generated images for ID #12



(b) Generated images for ID #14

(c) Generated images for ID #29



(d) Generated images for ID #70

Figure 7.6: Real images compared with generated images

7.6 Conclusion

In this chapter we introduced a brand new data set, HWD+, which contains almost 14 000 high-resolution images of hand-written digits attached to a set of labels containing the digit, the writer ID and various writer characteristics. The data set has been carefully collected and processed and is publicly available online.

We have done a first analysis of the data set; we shown that our data contains variables with different predictability making it a useful alternative to MNIST for testing new computer vision algorithms. We especially considered classification tasks that were made possible with our new data set such as including additional predictors in classification tasks or using higher-resolution images.

We have also proceed with a semi-supervised analysis. We have shown the potential use of our data set in a semi-supervised classification task in tandem with the MNIST data set; the use of the M2 model led to a more accurate LeNet-5 classifier. We have also shown the potential of our multi-label data set for controllable image generation.

We believe our data set is the perfect testing ground for new creative controllable generative models. Additionally, we would like investigate further the benefits of integrating MNIST for semi-supervised task given the positive results we have obtained so far.

Appendix

Biological Gender	Male 46	Female 51		
Handedness	Right 84	Left 13		
Language (education)	French 75	English 16	Other 6	
Education Level	No high school 7	High school 13	Bachelor 55	Graduate 22
Usual writing medium	Hand 44	Keyboard 45	Other 8	

Table 7.5: Table of occurence at the time of submission. The data set contains 97 writers.

Chapter 8

Conclusion

8.1 Summary

In this thesis we explored multiple facets of modern data analysis which relies more on algorithmic approaches every day. We began our work by visiting well-established techniques such as Decision Trees and Random Forests and ended up with modern computer vision algorithms such as VAEs and committees of CNNs.

In our first research project, we used a random forest to analyse a rich data set containing various student academic information. The data contained multiple predictors and because we believe that there might be high-order of interactions, a random forest was an appropriate model. Moreover the data set did not respect most of the linear analysis assumptions. We obtained multiple interesting results: we were able to both predict if a student succeeded and what would he major in using only the first year of courses and grades with a respectable accuracy. Additionally, we performed a variable importance analysis that revealed supporting evidence for the existence of grade inflation problem currently discussed in the higher education research literature.

However, to proceed with this analysis we had to deal with a data set with missing values; this inspired a follow-up project where we developed a new decision algorithm able to naturally manage missing values without any imputation. This new algorithm, coined BESTree, was tested against many simulated data sets and the motivating data set. Performances varied depending on the missingness structure but we demonstrated the great performances of BEST when the data is MAR or MNAR. Furthermore, BEST obtained the best performance compared to all other tested techniques on the real data set. We built a R-package, BESTree, which makes this new algorithm available to anyone who wishes to use it.

We then turned our attention to more modern statistical learning algorithms. One model that caught our attention was the Variational AutoEncoder. We are currently working on a project describing the current state of VAE implementations as there seems to be major differences between the theoretical model proposed and the successful implementations.

We have also built our own model using the VAE architecture to analyse a survival analysis data set. The data set contains information about young patients who suffered from Hodgkin-Lymphoma cancer.

We employed multiple machine learning survival analysis models and were forced to admit that most of these state-of-the-art models produced only slight improvements over the Cox Proportional Hazard model.

We finally dived into the field of computer vision. We collected a data set that offers multiple research opportunities and performed a first analysis. Using only a few images of digits, modern algorithms are able to accurately identify the writer of an unobserved digit. We were also able to showcase the potential generative problems approachable with the new data set. More precisely, we were able to build a VAE model that could generate new images of a selected digit that mimics the writer style of a selected writer.

8.2 Discussion and opinions

We noticed a complete shift in the way the research was produced when comparing the two eras of statistical learning: well-established models such as Random Forest and Support Vector Machine and very recent models. The publication focus shifted from journals to conferences which increases the speed of publication and has allowed for fast progress in many exciting research areas. On the other hand, reviewers are more than often given a few days to review multiple articles and this has reduced the quality of the average paper accepted. Bengio [14] recently published a blog post discussing the issues with conference publications.

This opens up a lot of opportunities for statisticians, probabilists and mathematicians to make theoretical contributions to the field of statistical and machine learning. There are also plenty of opportunities for applied statisticians since these new models are hard to interpret and hard to utilize to their full potential which makes knowledgeable data scientists more valuable.

The future of machine learning relies on its ability to adapt to a wide range of problems. Right now, machine learning techniques are extremely good for tasks like computer vision, recommendation systems and natural language processing. However, those algorithms have problems establishing themselves as solutions to well-studied statistical problems such as survival analysis, time series and spatial statistics. Surely we can find papers here and there that claim that one particular model shows higher accuracy in one of those tasks but there is no consensus yet and no well-establish machine learning model for many typical statistical problems. This again, creates multiple opportunity for the rigorous development on new statistical learning models suited for those tasks.

Above all, I believe the real difference between ML and statistics lies in the research approach. ML is much more concerned with implementations while statistics are concerned with concepts and models. In ML, the main contribution is usually an implementation and a paper with little to no algorithmic contribution might be rejected on sight. On the other hand, providing a concept and some theory supporting this concept is considered a contribution in statistics, so is a new analysis of a data set. This leads two different ways of solving problems: when facing a new data set most computer scientists will try to implement a new model until positive results while a statistician might be inspired by the data set to come with a new theoretically sound model without implementing it or actually solving the problem. Nonetheless these are generalizations, these fields are not monolithic blocks of some sort but rather large communities filled with different researchers with individual goals and dreams.

8.3 Future projects

At the moment, I want to follow up on my attempt to theoretically solve some of the VAE problems. I want to follow the lead of the variance identifiability problem; this would make a significant contribution to the VAE model. I am currently exploring a new measure for the gap between the data set's second moment and the second moment of the generative distribution obtained from training a VAE model. I have established a fast way to compute those two second moment estimators and to summarize this gap using matrix norm. This is the first step in my study of VAE's abilities to correctly fit the second moment a data set.

I would also like to invest some time on BESTree. To begin, I would like to integrate the new criteria proposed in Section 4.5.5 and update the R-package accordingly. Additionally, I would like to further optimize the R package by increasing its run speed using C++ (via Rcpp) when searching for the optimal partitioning.

Similarly, a few of my public contributions deserve an update. I would like to update the SAVAE code, comment it and make it public with examples and optimization insights. I would also like to update the HWD+ data set as I receive more data.

Finally, I desire to keep working on my long-term project which controllable content generation; I want to generalize generative models to allow them to improve the control the user has on the generative process. The HWD+ data set featured in Chapter 7 could be used for that particular purpose; we can try to generate an image while controlling both the digit and the writer style. Though I spent most of time working with VAEs in the last years, Generative Adversarial Networks (GANs) [56] is the dominant model for image generation and I want to get a grip of those in the coming projects.

Bibliography

[1] C. Adak, B. B. Chaudhuri, and M. Blumenstein. An empirical study on writer identification and verification from intra-variable individual handwriting. *IEEE Access*, 7:24738–24758, 2019.

[2] F. V. Agakov and D. Barber. An auxiliary variational method. In *International Conference on Neural Information Processing*, pages 561–566. Springer, 2004.

[3] A. Alemi, B. Poole, I.n Fischer, J. Dillon, R. A. Saurous, and K. P. Murphy. Fixing a broken ELBO. In *International Conference on Machine Learning*, pages 159–168, 2018.

[4] A. Asperti and M. Trentin. Balancing reconstruction error and Kullback-Leibler divergence in Variational Autoencoders. *arXiv preprint arXiv:2002.07514*, 2020.

[5] L. Aulck, N. Velagapudi, J. Blumenstock, and J. West. Predicting student dropout in higher education. *ArXiv e-prints*, June 2016.

[6] M. A. Bailey, J. S. Rosenthal, and A. H. Yoon. Grades and incentives: assessing competing grade point average measures and postgraduate outcomes. *Studies in Higher Education*, 41(9):1548–1562, 2016.

[7] T. Bar, V. Kadiyali, and A. Zussman. Grade Information and Grade Inflation: the Cornell experiment. *Journal of Economic Perspectivs*, 23(3):93–108, 2009.

[8] C. Beaulac. Hwd+ database. `https://drive.google.com/drive/folders/1f2o1kjXLvcxRgtmMMuDkA2PQ5Zato4Or`, 2020.

[9] C. Beaulac and J. S. Rosenthal. Predicting university students' academic success and major using random forests. *Research in Higher Education*, 60(7):1048–1064, 2019.

[10] C. Beaulac and J. S. Rosenthal. Analysis of a high-resolution hand-written digits data set with writer characteristics. *arXiv preprint arXiv:2011.07946*, 2020.

[11] C. Beaulac and J. S. Rosenthal. BEST: a decision tree algorithm that handles missing values. *Computational Statistics*, 35(3):1001–1026, 2020.

[12] C. Beaulac, J. S. Rosenthal, and D. Hodgson. A deep latent-variable model application to select treatment intensity in survival analysis. *Proceedings of the Machine Learning for Health (ML4H) Workshop at NeurIPS 2018*, 2018.

[13] C. Beaulac, J. S. Rosenthal, Q. Pei, D. Friedman, S. Wolden, and D. Hodgson. An evaluation of machine learning techniques to predict the outcome of children treated for Hodgkin-Lymphoma on the AHOD0031 trial. *Applied Artificial Intelligence*, 34(14):1100–1114, 2020.

[14] Y. Bengio. Time to rethink the publication process in machine learning. `https://yoshuabengio.org/2020/02/26/time-to-rethink-the-publication-process-in-machine-learning/`, Feb 2020.

[15] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[16] I. Bou-Hamad, De. Larocque, and H. Ben-Ameur. A review of survival trees. *Statistics Surveys*, 5, 01 2011.

[17] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[18] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[19] L. Breiman. Heuristics of instability and stabilization in model selection. *The Annals of Statistics*, 24(6):2350–2383, 12 1996.

[20] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[21] L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001.

[22] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[23] D. V. Budescu. Dominance analysis: a new approach to the problem of relative importance of predictors in multiple regression. *Psychological bulletin*, 114(3):542, 1993.

[24] C. P. Burgess, I. Higgins, A. Pal, L. Matthey, N. Watters, G. Desjardins, and A. Lerchner. Understanding disentangling in beta-VAE. *arXiv preprint arXiv:1804.03599*, 2018.

[25] S.-H. Cha and S. N. Srihari. Assessing the authorship confidence of handwritten items. In *Proceedings Fifth IEEE Workshop on Applications of Computer Vision*, pages 42–47. IEEE, 2000.

[26] H.-C. Chen, R. L. Kodell, K. F. Cheng, and J. J. Chen. Assessment of performance of survival prediction models for cancer prognosis. *BMC Medical Research Methodology*, 12(1):102, Jul 2012.

[27] R. Chen and S. L. DesJardins. Exploring the effects of financial aid on the gap in student dropout risks by income level. *Research in Higher Education*, 49(1):1–18, 2008.

[28] R. Chen and S. L. DesJardins. Investigating the impact of financial aid on student dropout risks: racial and ethnic differences. *The Journal of Higher Education*, 81(2):179–208, 2010.

[29] Y.-L. Chen, C.-L. Hsu, and S.-C. Chou. Constructing a multi-valued and multi-labeled decision tree. *Expert Systems with Applications*, 25(2):199 – 209, 2003.

[30] S. Chou and C.-L. Hsu. MMDT: A multi-valued and multi-labeled decision tree classifier for data mining. *Expert Systems with Applications*, 28(4):799–812, May 2005.

[31] D. C. Ciresan, U. Meier, L. Gambardella, and J. Schmidhuber. Deep, big, simple Neural Nets for handwritten digit recognition. *Neural Computation*, 22:3207–3220, 2010.

[32] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.

[33] A. Clare and R. D. King. *Knowledge discovery in multi-label phenotype data*, pages 42–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[34] D. R. Cox. Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(2):187–220, 1972.

[35] D. R. Cox. Partial likelihood. *Biometrika*, 62(2):269–276, 08 1975.

[36] B. C. Csáji. Approximation with artificial neural networks. *Faculty of Sciences, Etvs Lornd University, Hungary*, 24:48, 2001.

[37] B. Dai, Y. Wang, J. Aston, G. Hua, and D. Wipf. Hidden talents of the variational autoencoder. *arXiv preprint arXiv:1706.05148*, 2017.

[38] D. Decoste and B. Schölkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, March 2002.

[39] Y. Ding and J. S. Simonoff. An investigation of missing data methods for classification trees applied to binary response data. *Journal of Machine Learning Research*, 11:131–170, March 2010.

[40] C. Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[41] G. Dorta, S. Vicente, L. Agapito, N. D.F. Campbell, and I. Simpson. Structured uncertainty prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5477–5485, 2018.

[42] G. Dorta, S. Vicente, L. Agapito, N. D.F. Campbell, and I. Simpson. Training VAEs under structured residuals. *arXiv preprint arXiv:1804.01050*, 2018.

[43] D. Eddelbuettel and R. Francois. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(1):1–18, 2011.

[44] C. Evangelia, J. Ma, G. Collins, E. Steyerberg, J. Verbakel, and B. Van Calster. A systematic review shows no performance benefit of machine learning over logistic regression for clinical prediction models. *Journal of Clinical Epidemiology*, 110, 02 2019.

[45] A. J. Feelders. Handling missing data in trees: Surrogate splits or statistical imputation. In *Principles of Data Mining and Knowledge Discovery*, 1999.

[46] S. Fotso. Deep Neural Networks for Survival Analysis Based on a Multi-Task Framework. *arXiv e-prints*, page arXiv:1801.05512, Jan 2018.

[47] S. Fotso. PySurvival: Open source package for survival analysis modeling, 2019.

[48] D. L. Friedman, L. Chen, S. Wolden, A. Buxton, K. McCarten, T. J. FitzGerald, S. Kessel, P. A. De Alarcon, A. R. Chen, N. Kobrinsky, P. Ehrlich, R. E. Hutchison, L. S. Constine, and C. L. Schwartz. Dose-intensive response-based chemotherapy and radiation therapy for children and adolescents with newly diagnosed intermediate-risk Hodgkin Lymphoma: A report from the Children's Oncology Group Study AHOD0031. *Journal of Clinical Oncology*, 32(32):3651–3658, 2014. PMID: 25311218.

[49] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[50] J. Friedman, R. Kohavi, and Y. Yun. Lazy decision trees. 1, 09 1997.

[51] S. Gavankar and S. Sawarkar. Decision tree: Review of techniques for missing values at training, testing and compatibility. In *2015 3rd International Conference on Artificial Intelligence, Modelling and Simulation (AIMS)*, pages 122–126, Dec 2015.

[52] Y. Ge, R. Zhang, L. Wu, X. Wang, X. Tang, and P. Luo. DeepFashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5332–5340, 2019.

[53] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[54] J. Glaesser and B. Cooper. Gender, parental education, and ability: their interacting roles in predicting GCSE success. *Cambridge Journal of Education*, 42(4):463–480, 2012.

[55] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[57] E. Graf, C. Schmoor, W. Sauerbrei, and M. Schumacher. Assessment and comparison of prognostic classification schemes for survival data. *Statistics in medicine*, 18:2529–45, 09 1999.

[58] H. Haider. *MTLR: Survival Prediction with Multi-Task Logistic Regression*, 2019. R package version 0.2.1.

[59] D. J. Hand. Classifier technology and the illusion of progress. *Statistical Sciences*, 21(1):1–14, 2006.

[60] F.E. Harrell Jr, K.L. Lee, and D.B. Mark. Multivariable prognostic models: issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. *Statistics in medicine*, 15(4):361–387, 1996.

[61] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer, 2nd edition, 2009.

[62] J. He, D. Spokoyny, G. Neubig, and T. Berg-Kirkpatrick. Lagging inference networks and posterior collapse in variational autoencoders. *arXiv preprint arXiv:1901.05534*, 2019.

[63] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *ICLR*, 2(5):6, 2017.

[64] T. Hothorn, P. Bühlmann, S. Dudoit, A. Molinaro, and M. J. Van Der Laan. Survival ensembles. *Biostatistics*, 7(3):355–373, 12 2005.

[65] T. Hothorn, K. Hornik, C. Strobl, and A. Zeileis. *party: A Laboratory for Recursive Partytioning*, 2019. R package version 1.3-3.

[66] T. Hothorn, K. Hornik, and A. Zeileis. Unbiased recursive partitioning: A conditional inference framework. *Journal of Computational and Graphical Statistics*, 15(3):651–674, 2006.

[67] T. Hothorn, B. Lausen, A. Benner, and M. Radespiel-Tröger. Bagging survival trees. *Statistics in Medicine*, 23(1):77–91, 2004.

[68] H. Huang, R. He, Z. Sun, and T. Tan. Introvae: Introspective variational autoencoders for photographic image synthesis. In *Advances in neural information processing systems*, pages 52–63, 2018.

[69] H. Ishwaran and U. B. Kogalur. Consistency of random survival forests. *Statistics & Probability Letters*, 80(13):1056 – 1064, 2010.

[70] H. Ishwaran, U. B. Kogalur, E. H. Blackstone, and M. S. Lauer. Random survival forests. *The Annals of Applied Statistics*, 2(3):841–860, 09 2008.

[71] H. Ishwaran, U. B. Kogalur, E. Z. Gorodeski, A. J. Minn, and M. S. Lauer. High-dimensional variable selection for survival data. *Journal of the American Statistical Association*, 105(489):205–217, 2010.

[72] H. Ishwaran and U.B. Kogalur. *Fast Unified Random Forests for Survival, Regression, and Classification (RF-SRC)*, 2019. R package version 2.9.1.

[73] D. Jimenez Rezende, S. Mohamed, and D. Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. *ArXiv e-prints*, January 2014.

[74] B. Jinga, T. Zhangh, Z. Wanga, Y. Jina, K. Liua, W. Qiua, L. Kea, Y. Suna, C. Hea, D. Houh, L. Tanga, X. Lva, and C. Lia. A deep survival analysis method based on ranking. *Artificial Intelligence in Medicine*, 98:1 – 9, 2019.

[75] S. Randall Johnson and F. King Stage. Academic engagement and student success: Do high-impact practices mean higher graduation rates? *The Journal of Higher Education*, 0(0):1–29, 2018.

[76] V. E. Johnson. *Grade Inflation : A Crisis in College Education*. Springer, 2003.

[77] J. D. Kalbfleisch and R. L. Prentice. *The statistical analysis of failure time data*, volume 360. John Wiley & Sons, 2011.

[78] R. Kappe and H. van der Flier. Predicting academic success in higher education: what's more important than being smart? *European Journal of Psychology of Education*, 27(4):605–619, 2012.

[79] J. Katzman. DeepSurv: Personalized treatment recommender system using a Cox proportional hazards deep neural network, 2017.

[80] J. Katzman, U. Shaham, A. Cloninger, J. Bates, T. Jiang, and Y. Kluger. DeepSurv: Personalized treatment recommender system using a Cox proportional hazards deep neural network. *BMC Medical Research Methodology*, 18, 12 2018.

[81] D. Keysers, T. Deselaers, C. Gollan, and H. Ney. Deformation models for image recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1422–1435, 2007.

[82] H. Kim and W.-Y. Loh. Classification trees with unbiased multiway splits. *Journal of the American Statistical Association*, 96:589–604, 2001.

[83] D. P. Kingma. *Variational Inference & Deep Learning : A New Synthesis*. PhD thesis, Universiteit van Armsterdam, 10 2017.

[84] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[85] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling. Semi-supervised learning with deep generative models. In *Advances in neural information processing systems*, pages 3581–3589, 2014.

[86] D. P. Kingma and M. Welling. Auto-Encoding Variational Bayes. *ArXiv e-prints*, December 2013.

[87] R. J. Klimoski and A. Rafaelu. Inferring personal qualities through handwriting analysis. *Journal of Occupational Psychology*, 56(3):191–202, 1983.

[88] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

[89] I. Kononenko. On biases in estimating multi-valued attributes. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pages 1034–1040, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[90] M. Kuhn and R. Quinlan. *C50: C5.0 Decision Trees and Rule-Based Models*, 2018. R package version 0.1.2.

[91] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther. Autoencoding beyond pixels using a learned similarity metric. In *International conference on machine learning*, pages 1558–1566, 2016.

[92] M. LeBlanc and J. Crowley. A review of tree-based prognostic models. *Recent Advances in Clinical Trial Design and Analysis*, 75:113–124, 1995.

[93] M. E. Leblanc and J. P. Crowley. Relative risk trees for censored survival data. *Biometrics*, 48 2:411–25, 1992.

[94] Y. LeCun. Generalization and network design strategies. 1989.

[95] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[96] D. M. Leeds and S. L. DesJardins. The effect of merit aid on enrollment: A regression discontinuity analysis of iowa's national scholars award. *Research in Higher Education*, 56(5):471–495, Aug 2015.

[97] A. Liaw and M. Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002.

[98] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data*. John Wiley & Sons, Inc., 2002.

[99] Y. Liu, K. Gadepalli, M. Norouzi, G. E. Dahl, T. Kohlberger, A. Boyko, S. Venugopalan, A. Timofeev, P. Q. Nelson, G. S. Corrado, J. D. Hipp, L. Peng, and M. C. Stumpe. Detecting Cancer Metastases on Gigapixel Pathology Images. *arXiv e-prints*, page arXiv:1703.02442, Mar 2017.

[100] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[101] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[102] W.-Y. Loh. Regression trees with unbiased variable selection and interaction detection. *Statistica Sinica*, 12:361–386, 2002.

[103] W.-Y. Loh and Y.-S. Shih. Split selection methods for classification trees. *Statistica Sinica*, 7:815–840, 1997.

[104] C. Louizos, U. Shalit, J. Mooij, D. Sontag, R. Zemel, and M. Welling. Causal effect inference with deep latent-variable models. *ArXiv e-prints*, May 2017.

[105] J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi. Don't blame the ELBO! A linear VAE perspective on posterior collapse. In *Advances in Neural Information Processing Systems*, pages 9408–9418, 2019.

[106] M. Luck, T. Sylvain, H. Cardinal, A. Lodi, and Y. Bengio. Deep learning for patient-specific kidney graft survival analysis. *CoRR*, abs/1705.10245, 2017.

[107] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Improving semi-supervised learning with auxiliary deep generative models. In *NIPS Workshop on Advances in Approximate Bayesian Inference*, 2015.

[108] L. Maaløe, C. K. Sønderby, S. K. Sønderby, and O. Winther. Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*, 2016.

[109] J. S. Mills and K. R. Blankstein. Perfectionism, intrinsic vs extrinsic motivation, and motivated strategies for learning: a multidimensional analysis of university students. *Personality and Individual Differences*, 29(6):1191 – 1204, 2000.

[110] A. Nazábal, P. M. Olmos, Z. Ghahramani, and I. Valera. Handling incomplete heterogeneous data using VAEs. *ArXiv*, abs/1807.03653, 2018.

[111] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[112] A. S. M. Niessen, R. R. Meijer, and J. N. Tendeiro. Predicting performance in higher education using proximal predictors. *PLOS ONE*, 11(4):1–14, 04 2016.

[113] B. Ost. The role of peers and grades in determining major persistence in sciences. *Economics of Education Review*, 29:923–934, 2010.

[114] E. Park. Manifold learning with variational auto-encoder for medical image analysis. Technical report, Tech. Rep, 2015.

[115] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, S. Tejani, A.and Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[116] A. Peters and T. Hothorn. *ipred: Improved Predictors*, 2019. R package version 0.9-9.

[117] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[118] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013.

[119] G. Rahman and Z. Islam. Missing value imputation using decision trees and decision forests by splitting and merging records: Two novel techniques. *Knowledge-Based Systems*, 53:51 – 65, 2013.

[120] A. Rasmus, M. Berglund, M. Honkala, H. Valpola, and T. Raiko. Semi-supervised learning with ladder networks. In *Advances in neural information processing systems*, pages 3546–3554, 2015.

[121] R. Rastgoufard. Multi-label latent spaces with semi-supervised deep generative models. 2018.

[122] A. Rehman, S. Naz, and M. I. Razzak. Writer identification using machine learning approaches: A comprehensive review. *Multimedia Tools Appl.*, 78(8):10889–10931, April 2019.

[123] A. Rehman, S. Naz, M. I. Razzak, and I. A. Hameed. Automatic visual features for writer identification: A deep learning approach. *IEEE Access*, 7:17149–17157, 2019.

[124] A. Rodríguez-Ruiz, E. Krupinski, J.-J. Mordang, K. Schilling, S. H. Heywang-Köbrunner, I. Sechopoulos, and Ritse M. Mann. Detection of breast cancer with mammography: Effect of an artificial intelligence support system. *Radiology*, 290(2):305–314, 2019. PMID: 30457482.

[125] A. Rodriguez-Ruiz, K. Lång, A. Gubern-Merida, M. Broeders, G. Gennaro, P. Clauser, T. H. Helbich, M. Chevalier, T. Tan, T. Mertelmeier, M. G. Wallis, I. Andersson, S. Zackrisson, R. M. Mann, and I. Sechopoulos. Stand-Alone Artificial Intelligence for Breast Cancer Detection in Mammography: Comparison With 101 Radiologists. *JNCI: Journal of the National Cancer Institute*, 111(9):916–922, 03 2019.

[126] D. B. Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[127] D. E. Rumelhart, G. E. Hinton, and R. J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[128] O. Rybkin, K. Daniilidis, and S. Levine. Simple and effective VAE training with calibrated decoders. *arXiv preprint arXiv:2006.13202*, 2020.

[129] M. Saar-Tsechansky and F. Provost. Handling missing values when applying classification models. *Journal of Machine Learning Research*, 8:1623–1657, December 2007.

[130] R. Sabot and J. Wakeman-Linn. Grade inflation and course choice. *Journal of Economic Perspectives*, 5:159–170, 1991.

[131] J. L. Schafer and M. K. Olsen. Multiple imputation for multivariate missing-data problems: A data analyst's perspective. 33, 07 2000.

[132] S. Seaman, J. Galati, D. Jackson, and J. Carlin. What is meant by "missing at random"? *Statistical Science*, 28(2):257–268, 05 2013.

[133] S. Shalev-Shwartz and S. Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, New York, NY, USA, 2014.

[134] N. Sidiropoulos, S. H. Sohi, N. Rapin, and F. O. Bagger. SinaPlot: an enhanced chart for simple and truthful representation of single observations over multiple classes, 2017.

[135] N. Simon, J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for Cox's Proportional Hazards model via coordinate descent. *Journal of Statistical Software, Articles*, 39(5):1–13, 2011.

[136] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[137] C. K. Sønderby, T. Raiko, S. K. Maaløe, L.and Sønderby, and O. Winther. Ladder variational autoencoders. In *Advances in neural information processing systems*, pages 3738–3746, 2016.

[138] H. Steck, B. Krishnapuram, C. Dehing-Oberije, P. Lambin, and V. C. Raykar. On ranking in survival analysis: Bounds on the concordance index. In *Advances in Neural Information Processing Systems 20*, pages 1209–1216. Curran Associates, Inc., 2008.

[139] C. Strobl, A.-L. Boulesteix, A. Zeileis, and T. Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics*, 8(1):25, 2007.

[140] T. Therneau. *A Package for Survival Analysis in R*, 2021. R package version 3.2-10.

[141] T. Therneau and B. Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2018. R package version 4.1-13.

[142] N. J. Tierney, F. A. Harden, M. J. Harden, and K. L. Mengersen. Using decision trees to understand structure in missing data. *BMJ Open*, 5(6), 2015.

[143] M. E. Tipping and C. M. Bishop. Mixtures of probabilistic principal component analyzers. *Neural computation*, 11(2):443–482, 1999.

[144] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 61(3):611–622, 1999.

[145] J. Townsend, T. Bird, and D. Barber. Practical lossless compression with latent variables using bits back coding. *arXiv preprint arXiv:1901.04866*, 2019.

[146] B. Twala. An empirical comparison of techniques for handling incomplete data using decision trees. *Applied Artificial Intelligence*, 23(5):373–405, May 2009.

[147] B. Twala, M.C. Jones, and D. Hand. Good methods for coping with missing data in decision trees. 29:950–956, 05 2008.

[148] University of Toronto. Degree requirements (h.b.a., h.b.sc., bcom), 2017.

[149] S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011.

[150] G. Van Rossum and F. L Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[151] L. Wen, Y. Zhou, L. He, M. Zhou, and Z. Xu. Mutual information gradient estimation for representation learning. *arXiv preprint arXiv:2005.01123*, 2020.

[152] H. Wickham. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016.

[153] Y.-J. Xiong, Y. Lu, and P. S. P. Wang. Off-line text-independent writer recognition: A survey. *International Journal of Pattern Recognition and Artificial Intelligence*, 31(05):1756008, 2017.

[154] C.-N. Yu, R. Greiner, Hsiu-Chin Lin, and V. Baracos. Learning patient-specific cancer survival distributions as a sequence of dependent regressors. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 1845–1853. Curran Associates, Inc., 2011.

[155] Y. Zhang and Z. Lu. Exploring semi-supervised variational autoencoders for biomedical relation extraction. *Methods*, 166:112–119, 2019.

[156] L. Zhao and D. Feng. DNNSurv: Deep neural networks for survival analysis using pseudo values. *arXiv e-prints*, page arXiv:1908.02337, Aug 2019.