UCLouvain

École polytechnique de Louvain

An investigation of the Information Bottleneck (IB) method in supervised learning

and how decision trees can be represented in the IB framework

Author: **Alexandre TYTGAT**
Supervisor: **Jean-Charles DELVENNE**
Readers: **Michel VERLEYSEN, Marco SAERENS**
Academic year 2020–2021
Master [120] in Data Science: Information technology

Abstract

The last two decades have been marked by rapid progress in the fields of machine learning and data science, for which the state of the art has continued to push new frontiers previously thought to be decades away. While remarkable, many of these powerful techniques have important shortcomings such as a high energy usage, risks of bias, and for some a deep lack of theoretical understanding. The Information Bottleneck (IB) method has been proposed as a novel explanation of supervised learning. Originally introduced in the context of Information theory as a principled approach to extract relevant information in a signal, this method has since found a wide variety of applications beyond its initial purpose. In this work, we study the capabilities of the IB framework to evaluate important aspects of supervised models in the important case of classification tasks. Our results indicate that the IB framework is insufficient to adequately evaluate both the quality of a model's fit to the training data, as well as its ability to generalize to unseen observations. Moreover, we also present the first to our knowledge analysis of decision trees in the IB framework.

Acknowledgments

First, I would like to thank professor Jean-Charle Delvenne for his open mindedness, his precious time, and his unwavering optimism during the course of this work.

To my closest friends. Thank you for staying by my sides during the good and the tough times. Know that I cherish your friendship thoroughly, for you are one of most valuable part of my life.

To my parents. You gave me more I could have hoped for. For that and your unconditional love, I am eternally grateful to you.

Carl. You are the closest thing there is to a soulmate to me. I love you from the bottom of my heart.

Finally, to the wonderful people that I met in the Cyclotron almost 6 years ago, to the immensely funny geography students that I had the chance to befriend, to my beloved roommates who made me love life more than ever before, to my tennis buddies and the good (and sweaty) people of the cardio fit, to you my love for all of you are and all that you have done for me, to my family for each one of you is an integral part of my life, to the incredibly nice and intelligent bio-engineering students with whom I have spend memorable moments, to my data brothers, to Kate for her precious help in proofreading this work, to everyone from the TO3 scouting unit who accepted me and made me feel one of their own, to the great teachers that I had the chance to learn from at university, to the few who have been by side for more years than I can count, and to the ones whom I got to know more recently.

Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The last two decades have been marked by rapid progress in the fields of machine learning and data science, for which the state of the art has continued to push new frontiers previously thought to be decades away. The variety of their domains of applications has made these techniques an essential part in a multitude of industries, but also as tools of choice in numerous domains of scientific research.

While remarkable, many of these advances have important shortcomings. They use a tremendous amount of computational power, and as such have high energy costs which is undesirable for a society that needs to limit its carbon footprint. Moreover, they can suffer from undesirable bias that can impact individuals, raising ethical questions about their usage. Finally, popular methods such as deep learning suffers from a lack of theoretical understanding, which in turn slows down their development.

A common aspect of these methods is their ability to learn from experience. Thus, understanding their learning process is essential to guide new improvements which could reduce their energy needs and provide insights into how to minimize their bias.

In this regard, the Information Bottleneck (IB) has been widely discussed as a novel explanation of supervised learning. Originally proposed in the context of Information theory as a principled approach to extract relevant information in a signal, this method has since found a wide variety of applications beyond its initial purpose.

In this work, we offer a review of this theory along with new insights regarding the important case of supervised classification. Specifically, we will provide supporting evidence for the usage of the IB to evaluate a model fit to the data, and we will also show that the IB framework is generally insufficient to evaluate the generalization capabilities of a model. Moreover, we will also present the first application to our knowledge of an IB analysis to decision trees, which are our model of choice to study empirically the scenarios considered in our theoretical analysis.

The structure of this work is as follows. Chapter 2 will introduce the general Information Bottleneck framework, which has application beyond learning. It will be explained how optimal representations are defined in the IB sense. We will also present the *information plane*, which is a diagram used to evaluate the quality of representations. Then, the first part of Chapter 3 will cover supervised learning along with the IB theory of learning. The second part of this chapter will be dedicated to the presentation of our theoretical analysis of the ability of the IB framework to assess the performances of models in the case of classification tasks. Next, decision trees will be presented in detail in Chapter 4, as they play an essential role in the following chapter to conduct our experiments. Finally, Chapter 5 will start by a presentation on the way decision trees can be interpreted in the IB framework, followed by a showcase of experiments conducted to justify our previous analysis.

Chapter 2

The Information Bottleneck framework

In this chapter, we present the general IB framework, from its origin in Information Theory, to the recent advancement made to solve the IB problem.

The bottleneck framework has been linked to many fields including supervised learning [2, 6], unsupervised learning [6], variational autoencoders [7], representation learning[8], remote source coding [2], minimal sufficient statistics [6], linguistics [9], and neuroscience [10]. Moreover, it has found numerous applications such as in the development of new deep learning architectures [7, 11–17], clustering techniques [18–20], and computational linguistics [21].

In this chapter, the Information Bottleneck method is introduced, along with all the central notions related to it. First, Section 2.1 will give a remainder of information measures such as the entropy and the mutual information, as they are central in the IB framework. Then, Section 2.2 will establish the Information Bottleneck problem in detail. After that, Section 2.3 will introduce the *IB curve* as the set of optimal bottleneck representations, along with the *information plane* in which the IB curve is represented. Finally, common methods to solve the IB problem will be presented in Section 2.4, and a summary of the chapter will be offered in Section 2.5.

2.1 Information measures

Before diving straight into the IB, we need to recall a few notions from Shannon's information theory, namely the concepts of entropy, conditional entropy and mutual information.

The *entropy* was introduced by Claude Shannon in [22] as a measure of the average information, or uncertainty, in a random variable outcome. Given a

discrete random variable X with support set $\mathcal{X}$ and probability distribution $p(X)$, the entropy of X is defined as

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{2.1}$$

where the logarithm is in base two. Here, we adopt the notation $p(x) = p(X = x)$ for the probability of a realization x. The entropy satisfies the following properties,

(i) It is positive : $H(X) \geq 0$.

(ii) It is maximal for uniformly distributed random variables (maximal uncertainty in the outcome).

(iii) If the probability distribution is deterministic, then the entropy is null $H(X) = 0$, because there is no uncertainty in the outcome of X.

By analogy with the discrete expression given in Equation 2.1, it is possible to extend the concept of entropy to continuous random variable by simply replacing the sum symbol with an integral

$$H(X) = -\int_{\mathcal{X}} dx \; p(x) \log p(x) \tag{2.2}$$

where $p(X)$ is now to be interpreted as the probability *density* function of X. This quantity is sometimes referred to as the *differential entropy*.

The concept of entropy has also been extended to random variables conditioned on another one. Given another random variable Y with support set $\mathcal{Y}$ and probability distribution $p(Y)$, the *conditional entropy* is defined as

$$\begin{aligned} H(X|Y) &= \sum_{y \in \mathcal{Y}} p(y) H(X|y) \\ &= -\sum_{y \in \mathcal{Y}} p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y) \\ &= -\sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x, y) \log p(x|y). \end{aligned} \tag{2.3}$$

It can be interpreted as the average amount of information left in the random variable X when the outcome of Y is known. It has the following properties :

(i) For any X and Y, the conditional entropy is always lower than the entropy of X : $H(X|Y) \leq H(X)$.
Intuitively, if the outcome of Y gives some information about X, then there should be less uncertainty about X than if the outcome is not known. On the other hand, if the outcome of Y does not give any information about X, then the amount of uncertainty about X would remain the same whether the outcome is known or not.

(ii) If X and Y are independent, then the conditional entropy is equal to the entropy of X : $H(X|Y) = H(X)$.
Simply generalizing the above argument, if all the possible outcomes of Y do not yield any information about Y, then the level of uncertainty about X would remain the same.

(iii) Finally, if the realisation of X is determined by the outcome of Y, then $H(X|Y) = 0$ since there is no uncertainty left if the value of Y is given.

Again it is possible to define a continuous version using the same analogy, called the *conditional differential entropy*

$$H(X|Y) = \int_{\mathcal{Y}} \int_{\mathcal{X}} dydx \; p(x,y) \log p(x|y). \tag{2.4}$$

If one variable is discrete and the other continuous, then the conditional entropy is given by a mixed expression. For instance, if Y is continuous and X is discrete then the conditional entropy is expressed by

$$\begin{aligned} H(X|Y) &= \int_{\mathcal{Y}} dy \; p(y) H(X|y) \\ &= \int_{\mathcal{Y}} dy \; p(y) \sum_{x \in \mathcal{X}} p(x|y) \log p(x|y) \end{aligned} \tag{2.5}$$

where $p(Y)$ is the probability density distribution of Y and $p(X|Y)$ is the probability mass distribution of X given Y.

For simplicity sake, the terms entropy and conditional entropy will be used to refer to the discrete, continuous and mixed version unless specified otherwise.

Finally, let us define the *mutual information* as a measure of the dependence between two random variables. It can be defined as,

$$\begin{aligned} I(X;Y) &= H(X) - H(X|Y) \\ &= H(Y) - H(Y|X). \end{aligned} \tag{2.6}$$

In the language of information theory, it corresponds to the shared information between X and Y or equivalently, the amount of information obtained about a random variable by observing the other random variable. It has the following properties :

(i) It is positive : $I(X;Y) \geq 0$.

(ii) It is symmetric : $I(X;Y) = I(Y;X)$.

(iii) If there exists a deterministic relationship $f(X) = Y$ between the two variables, then by the third property of the conditional entropy, $H(Y|X) = 0$, and so $I(X;Y) = H(Y)$.

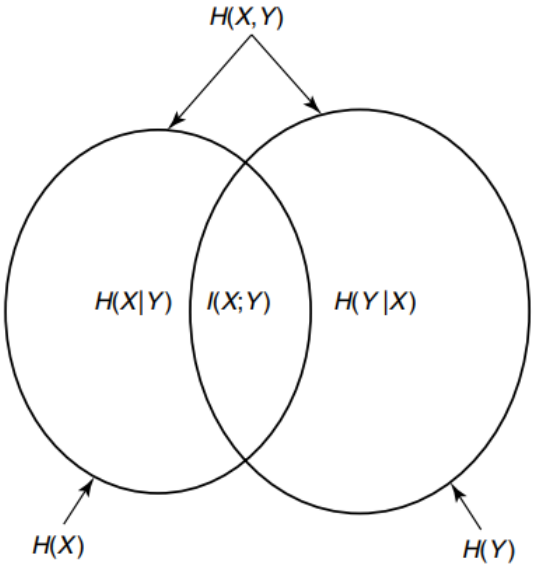


Figure 2.1: *Relationships between the different information quantities [1].*

(iv) If X and Y are independent, then $H(Y|X) = H(Y)$. Hence, $I(X;Y) = 0$ indicating that no information can be obtained about Y from X and vice-versa.

(v) It is invariant under any invertible transformation f : $I(X;Y) = I(f(X);Y)$.

Figure 2.1 summarizes the relationships between all the described information measures.

2.2 The IB method

Let us now introduce the IB method. Consider two random variables X and Y with some statistical dependency, i.e. strictly positive mutual information. The IB method is an approach to find a maximally compressed representation T obtained from X that preserves as much as possible the *relevant information* about Y, quantified by the mutual information term $I(X;Y)$.

The IB method further assumes that the information contained in bottleneck representation is solely extracted from the variable X. Formally, this can be expressed by the Markov chain $T - X - Y$, which formalizes the conditional independence of Y and T given the input X, i.e. $H(Y|T,X) = H(Y|X)$. This directly translates to the following factorization of the joint distribution of the three random variables,

$$p(X,Y,T) = p(T|X)p(Y|X)p(X). \tag{2.7}$$

The random variables that satisfy this chain must also satisfy the data-processing inequality (Theorem 2.8.1 in [1]) given by

$$I(X;T) \geq I(Y;T). \tag{2.8}$$

This inequality represents the flow of information between the three variables.

Another key ingredient to the IB method is how the compression and the relevance are measured. It is proposed that :

- The *relevance* or *accuracy* of a representation be measured by $I(Y;T)$.
- The *compression* or *complexity* of a representation be measured by $I(T;X)$.

These choices are motivated by their usage in Rate Distortion Theory, where optimal compression is obtained by the minimization of mutual information, and Noisy Channel Coding in which its maximization corresponds to optimal information transmission. For these reasons, these measures provide a natural approach to quantify the relevance or the meaningfulness of intermediate representations.

Other measures have been proposed, resulting in different formulations of the IB method. See Appendix A for a summary of three IB variations that have been suggested in the literature.

Putting together all the above elements, the IB can be formulated as a constrained non-linear optimization problem given by the so called *IB-functional*

$$F_{\text{IB,max}}(r) = \max_{T \in \Delta} I(Y;T) \quad \text{s.t.} \quad I(X;T) \leq r, \tag{2.9}$$

where Δ is the set of all possible bottleneck representations that satisfy the Markov condition $T - X - Y$, and $r \geq 0$ is an adjustable parameter that controls the level of compression desired. Lower values of r favors more compressed representation whereas larger values favors representations with a larger amount of relevant information.

Let us examine two limiting cases, one where $r = 0$ and the other for arbitrarily large values of r.

(i) **Maximal compression** : If the compression parameter is set to $r = 0$, then the mutual information $I(X;T)$ would also be zero, and the compression of the representation would be maximal. All realisations of X would be mapped to a unique point, and T would be completely uninformative about both X and Y. In this case, both the compression and the relevance measures would be equal to zero : $I(X;T) = I(Y;T) = 0$.

(ii) **No compression** : If $r \to \infty$, then the constraint on the compression term vanishes, and the optimal IB representation is simply X. Hence, this solution satisfies $I(X;T) = H(X)$.

In practice, the IB problem formulation given in Equation 2.9 is almost never used because it is too complex to solve. It is often preferred to solve a relaxed version by maximizing the so-called *IB-Lagrangian* over Δ [3, 6, 7, 23–26],

$$\mathcal{L}_{\text{IB}}^{\beta}(T) = I(Y;T) - \beta I(X;T) \tag{2.10}$$

where β is the Lagrange multiplier controlling the trade-off between relevance and compression. Contrary to the parameter r, lower values of β favors representations more informative about Y, and larger values favors more compressed representations.

Note that the trade-off parameter can be restricted to the interval $\beta \in [0, 1]$. This can be seen by examining the cases where $\beta \leq 0$ and $\beta \geq 1$.

(i) **Maximal compression** : For any $\beta \geq 1$, the data-processing inequality, Equation 2.8, implies that $\mathcal{L}_{\text{IB}}^{\beta}(T) \leq 0$. Thus, the optimal solution must solve $I(X;T) = I(T;Y) = 0$, which corresponds to the case of maximal compression $r = 0$. So the value $\beta = 1$ can be used as an upper bound since it corresponds to the same solutions as all trade-off values strictly above it.

(ii) **No compression** : If $\beta \leq 0$, then the sign of the compression term in the IB-Lagrangian is positive and the constraint vanishes like it does when the parameter r is unbounded. Hence, the solution for any negative values of β is X, so we can simply bound β from below by zero.

For information purposes only, the formulation of the IB problem in Equation 2.9 is not the only one used in the literature. Some authors prefer to work with the dual constrained optimization problem given by

$$F_{\text{IB,min}}(i) = \min_{T \in \Delta} I(X;T) \quad \text{s.t.} \quad I(Y;T) \geq i, \quad \text{with } i \geq 0 \tag{2.11}$$

and its corresponding dual IB-Lagrangian

$$\mathcal{L}_{\text{IB,dual}}^{\beta}(T) = I(X;T) - \beta^{-1} I(Y;T). \tag{2.12}$$

Both formulations have been shown to be equivalent in [24], but we shall only consider the former throughout this work.

2.3 The information plane and the IB curve

A particularly insightful observation about the IB-functional 2.9 is that it is a concave function of the parameter r [24, 27]. In the IB framework, this function

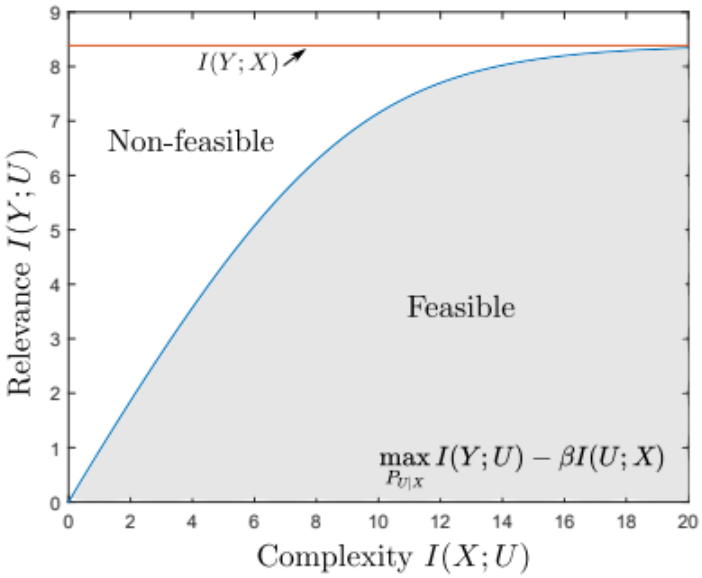


Figure 2.2: *The information plane is a central element of the IB framework. Any representation can be characterized by its information content in this diagram. [2].*

is commonly referred to as the *IB curve*, and it is often represented in a two-dimensional space, the so-called *information plane*, defined by vertical axis $I_Y = I(Y;T)$ and horizontal axis $I_X = I(T;X)$.

Any representation $T \in \Delta$ can be specified in the information plane by its information content given by the coordinates (I_X, I_Y). In particular, the IB optimal solutions all belong to the IB curve by definition. Interestingly, the IB curve delimits the information plane into two regions. Coordinates below the IB curve correspond to achievable representations, i.e. any $T \in \Delta$, and coordinates above the IB curve are unachievable by any bottleneck representations. Figure 2.2 shows the information plane with a typical IB curve.

The IB curve is subject to natural bounds such as a limit on the information content about Y that the representation T can possess. At some point, as r increases, the value of I_Y saturates at $I(Y;X)$ since no more information about Y can be extracted from X. The IB curve is also bounded at $I_X = H(X)$ because the representation can not be more complex than X itself. A third bound is given by the data-processing inequality Equation 2.8 which states that $I_Y \leq I_X$, and so limits the amount of information about Y that the representation can capture at any point.

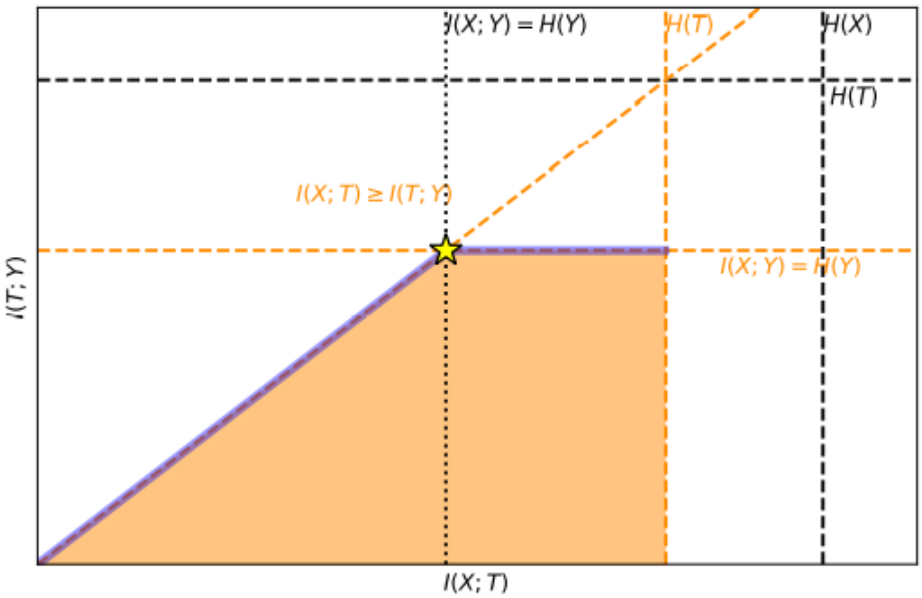


Figure 2.3: *The IB curve saturates several bounds when Y is obtained by a deterministic map from X [3].*

Deterministic IB curve

As pointed out in [28], the IB curve might not be strictly concave. In particular, it was shown that when the variable Y is fully determined by X, meaning that there exists a deterministic map $f(X) = Y$, the IB curve saturates the first and the third bound given by $I_Y \leq I(X;Y)$[1], and $I_Y \leq I_X$ respectively. Hence, for deterministic prediction task where such maps exist, the IB curve is piecewise linear as shown on Figure 2.3. The coordinates of the point where the two linear curves are joined - its *elbow* - elbow are $I_Y = H(Y)$ and $I_X = H(Y)$ exactly. This IB solution corresponds to the representation $T = f(X) = Y$.

In such cases, it was also proved that the IB-Lagrangian can not recover all points of the IB curve by varying β. Specifically, values of $\beta \in [0, 1]$ solely yields the IB solutions corresponding to the elbow point. Some variations of the IB problem have been shown to possess Lagrangians able to recover all points of the IB curve. See Appendix A for more details.

These properties of the IB problem will be central to our analysis of the IB theory of learning in the next chapter.

[1] In this case, the relevant information is equal to the entropy of Y since it is entirely determined by X (see property (iii) of the mutual information in Section 2.1)

2.4 Solutions

The problem of finding the IB optimal representation T is equivalent to the one of finding the optimal encoding and decoding maps $p(t|x)$ and $p(y|t)$ respectively. It follows from the joint distribution factorization in Equation 2.7 that, for continuous valued X, the decoding map is given by[2]

$$\begin{aligned} p(y|t) &= \frac{\int_{\mathcal{X}} dx\ p(y|x)p(t|x)p(x)}{p(t)} \\ &= \frac{\int_{\mathcal{X}} dx\ p(y|x)p(t|x)p(x)}{\int_{\mathcal{X}} dx\ p(t|x)p(x)}. \end{aligned}$$

If the joint distribution $p(X, Y)$ is known, then the problem of finding the optimal representation is reduced to retrieving the optimal encoding map $p(t|x)$.

In general, the optimization of the IB-Lagrangian is challenging because (i) the objective is not convex so there is no guarantee to reach a global optimum, (ii) even finding a local optimum requires the computation of mutual information terms which can involve intractable integrals. Notably, it is unfeasible to solve exactly for general continuous distribution $p(x, y)$. The only known continuous case where an analytical solution has been found is for jointly gaussian distributions [26], where the encoder and decoder are linear maps (see Appendix B for a more detailed presentation of this solution).

For discrete X and Y, under the assumption that the joint distribution $p(X, Y)$ is known, the solution which maximizes the IB-Lagrangian for a given β value was derived in [23], and is given by the set of self-consistent equations,

$$\begin{cases} p(t|x) = \frac{p(t)}{Z(x,\beta)} \exp\left[-\beta^{-1} D_{KL}(p(y|x)||p(y|t)\right] \\ p(t) = \sum_{x\in\mathcal{X}} p(t|x)p(x) \\ p(y|t) = \sum_{x\in\mathcal{X}} p(y|x)p(x|t) \end{cases} \tag{2.13}$$

where $Z(x, \beta)$ is a constant factor which ensures the normalization of the distributions. The symbol D_{KL} appearing in the first equation denotes the Kullback–Leibler divergence. It is a measure of how much the two distributions are alike and it is defined for two discrete distributions $p(Y)$ and $q(Y)$ by

$$D_{KL}(p||q) = \sum_{y\in\mathcal{Y}} p(y) \log \frac{p(y)}{q(y)}.$$

Note that the solution given in Equation 2.13 is formal since the encoding map appears implicitly in the exponential via the decoding map. This problem is usually

[2]For discrete X, the decoding map expression is the same but with sums running over all possible values of X instead of a integrals.

solved with the Blahut-Arimoto algorithm [29], though it does not guarantee a global solution. Moreover, it is computationally expensive for big state space.

Fortunately, recent progress have been made to solve the IB problem in more general settings with either deep learning-based methods [7, 11, 13, 14] or kernel-based methods [12]. They provide approximate solutions for situations in which the joint distribution is unknown and only samples are available. The common approach is to approximate a lower bound to the IB-Lagrangian and use it as the objective to be maximized. Moreover, some assumptions are often made on the type of encoding and/or decoding maps $p(T|X), p(Y|T)$. Even though these techniques have widened to a great extend the tasks where the IB method can be applied, [28] lists the following difficulties that these methods can be subject to when trying to find IB solutions :

- Difficulty of optimizing the IB objective.
- Error in estimating the mutual information terms.
- Mismatch between the assumptions made on the form of the encoding/decoding maps and the optimal ones.
- Stochasticity of training due to the stochastic gradient procedure of neural networks.

2.5 Summary

In this chapter, we presented the essential notions pertaining to the Information Bottleneck (IB) framework. Here, we offer a recap of the main points :

- The IB method is an information theoretic approach to obtain maximally compressed representation T of a random variable X that are the most informative about another random variable Y. Such representations are sometimes referred to as *bottleneck variables* or *bottleneck representations*.
- The IB is formally posed as a maximization problem of the mutual information term $I(Y;T)$ - the *relevance* of the representation - with a constraint on the level of compression given by $I(T;X)$) - the *complexity* of the representation.
- Representations are characterized by their information content $I_Y = I(Y;T)$ and $I_X = I(T;X)$, and are represented in the *information plane*, which is the diagram defined by the horizontal and vertical axis I_X and I_Y. In particular, the coordinates of the set of optimal representations form a concave curve in the information plane, referred to as the *IB curve*.

- For tasks where Y is fully determined by X, the IB curve is piecewise linear and is fully determined by the entropy of Y, which corresponds to the coordinates of the elbow of the curve. Moreover, this point designates the representation equal to Y.
- Solutions to the IB problem are generally obtained via a relaxed version of the IB problem, defined by the *IB-Lagrangian*. Several methods exist to solve it depending on the task. Exact solutions only exists for specific cases, e.g. a gaussian joint distribution. Often, only local solutions can be obtained or approximations to those.

The following chapter will cover the recent attempts made to understand supervised learning through the Information Bottleneck framework, along with new contributions to this topic in the important case of classification tasks.

Chapter 3

The IB theory of learning

Now that the IB framework has been set in the previous chapter, we are able to introduce the IB theory of learning, a proposed new look at supervised learning through the lens of information. As stated in the introduction, we will also provide new insights into this theory, in the case of deterministic classification tasks.

First, Section 3.1 will set the framework of statistical learning theory in which supervised learning is commonly understood. Then, Section 3.2 will establish the current state of the IB theory of supervised learning. It will be shown that it is yet to be proven as a solid framework able to encompass essential aspects of learning. After that, an original analysis on the capability of the bottleneck framework to assess the quality of a model to fit the data and its ability to generalize will be presented in Section 3.3. Finally, Section 3.4 will conclude with a summary of the chapter.

3.1 Supervised learning

Supervised learning is a subfield of machine learning in which algorithms, characterized by a set of tuning parameters $\boldsymbol{\theta}$[1], use a labeled dataset $S = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$ to learn a prediction map $f_{\boldsymbol{\theta}}$ between the input vectors $\boldsymbol{x}$ and their associated output value y. The values of the parameters $\boldsymbol{\theta}$ define the model trained on S, and are generally interpreted as a measure of its complexity. It is commonly assumed that the observations in S are independent and identically distributed from a joint, often unknown, probability distribution $p(X, Y)$ associated to the random variable (X, Y) with support set $\mathcal{X} \times \mathcal{Y}$. When the target variable Y is continuous-valued the supervised task is called *regression*, and *classification* when it is discrete.

The quality of a model predictions are evaluated using a *Loss function* $L(Y, f_{\boldsymbol{\theta}}(X))$, which measures the discrepancy between the observed values of Y and the predicted

[1] Sometimes called the *hyperparameters*

values $f_{\boldsymbol{\theta}}(X)$. Common measures include :

- For regression,

$$L(Y, f_{\boldsymbol{\theta}}(X)) = \begin{cases} (Y - f_{\boldsymbol{\theta}}(X))^2 & \text{(squared error)} \\ |Y - f_{\boldsymbol{\theta}}(X)| & \text{(absolute error).} \end{cases}$$

- For classification,

$$L(Y, f_{\boldsymbol{\theta}}(X)) = \begin{cases} I(Y \neq f_{\boldsymbol{\theta}}(X)) & \text{(0-1 loss)} \\ -2 \log \hat{p}_{\boldsymbol{\theta}}(y_k|X) & (-2 \times \text{ log-likelihood}) \end{cases}$$

  where I denotes the indicator function which is equal to 0 if $Y = f_{\boldsymbol{\theta}}(X)$ and 0 otherwise. The estimated probability $p_{\boldsymbol{\theta}}(y_k|X)$ corresponds to the frequency that the model $f_{\boldsymbol{\theta}}$ predicts the class y_k given X.

The *generalization error* or the *test error* is defined as the expectation of the loss function for a prediction map obtained with a specific training set S[4],

$$\text{gen}(\boldsymbol{\theta}; S) = \mathbb{E}[L(Y, f_{\boldsymbol{\theta}}(X)|S]$$

Since different training sets might yield different prediction maps, with higher or lower generalization errors, it is desirable to remove this excessive randomness by averaging out the training sets S, therefore we consider the *expected generalization error* or the *expected test error* defined by

$$\text{gen}(\boldsymbol{\theta}) = \mathbb{E}[\text{gen}(\boldsymbol{\theta}; S)] = \mathbb{E}[L(Y, f_{\boldsymbol{\theta}}(X)].$$

It is the goal of supervised learning to find the parameters $\boldsymbol{\theta}$ that minimize this quantity.

Figure 3.1 illustrates the generalization errors obtained with several training sets sampled from $p(X, Y)$ (light red lines), and the expected generalization error (solid red line) which is estimated by averaging the former. The horizontal axis corresponds to the complexity of the model which depends on the supervised method used, but for which the measure often involves the parameters $\boldsymbol{\theta}$

In most applications, supervised learning methods are trained to minimize the *training error*, which is the average loss over the training set S,

$$\bar{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y^{(i)}, f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})).$$

However, the training error is generally a bad estimate of the expected generalization error, as can be seen on Figure 3.1 where the training errors of models trained

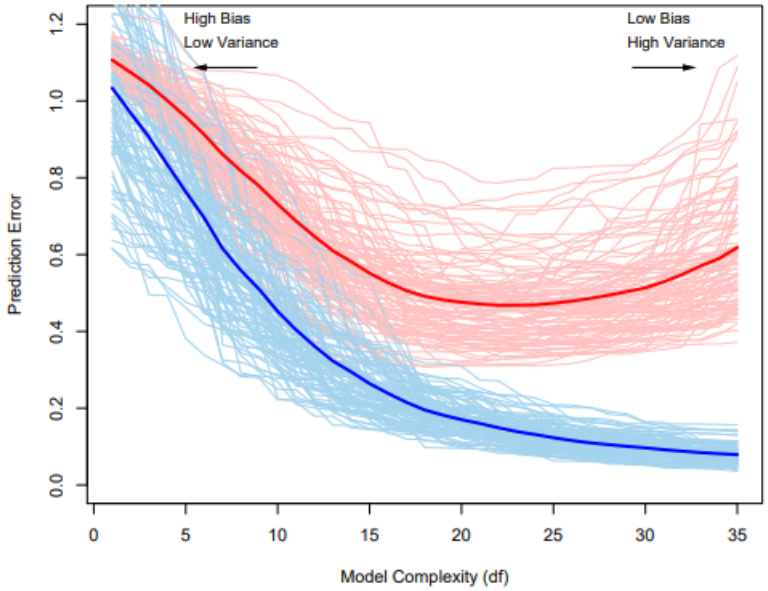


Figure 3.1: *Learning curves. The goal of any supervised model is to reach the minimum of the average test error (solid red line) [4].*

on different training sets are shown as the light blue curves and the solid blue curve corresponds to their average. More complex models are able to construct more elaborate prediction maps $f_{\boldsymbol{\theta}}$ which often adapt too much to the training data. Hence, the training error generally decreases with the model complexity, and sometimes to the expense of increasing the generalization error. This phenomenon is referred to as *overfitting*, and it occurs as soon as the generalization error starts to increase with the model complexity, even though the training error keeps decreasing. On Figure 3.1, we observe that all models with a complexity measure above $\sim$20 overfit. On the other hand, all models before this point are also suboptimal since their generalization error is not the lowest possible. These models under-perform because they are not complex enough, and are said to *underfit* because their limited complexity does not allow them to capture the full underlying pattern in the data in order to generalize correctly.

In order to limit overfitting, a widely used approach is to add a penalty term on the complexity of the model to the loss function, and to look for the parameters $\boldsymbol{\theta}$ that minimize this functional. This type of technique is called *regularization*, and the objective function often takes the following form,

$$\arg\min_{\boldsymbol{\theta}} \sum_{i=1}^{N} L(y^{(i)}, f_{\boldsymbol{\theta}}(\boldsymbol{x}^{(i)})) + \lambda J(\boldsymbol{\theta}). \tag{3.1}$$

where $J(\boldsymbol{\theta})$ is the *regularizer* and λ is a real parameter which controls the importance

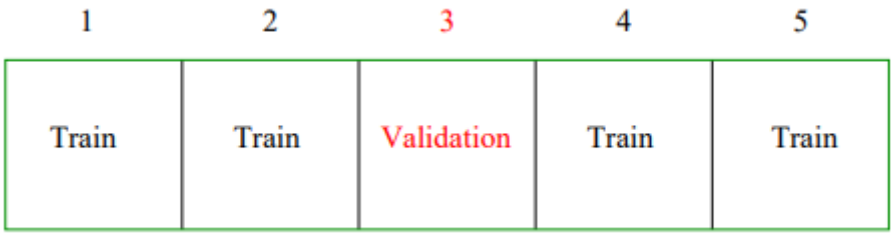


Figure 3.2: *An example of a training set divided into 5 folds used for the cross-validation [4].*

Figure 3.3: *The training set is used to fit the models, whereas the validation set is used to estimate prediction error for model selection, and finally the test set is used for evaluating the generalization error of the best performing model [4].*

of the penalty term.

One of the most common methods used to estimate the expected test error is the *K-fold cross validation*, or *K-fold CV* for short. It works by repeatedly fitting a model on K different training sets and evaluating each time the test error on a separate set of unseen observations. It does so by dividing the set S into K folds as shown on Figure 3.2, fitting a model on a training set consisting of $K-1$ folds, and evaluating the test error on the remaining fold, named the *validation set*. Then, it computes the estimated expected test error by taking the average of the K values for the test error obtained during the procedure. Using this procedure, it is possible to evaluate adequately the generalization performance of different models, and to select the one with the lowest estimated expected test error.

Furthermore, it is often the case that the final model selected is evaluated on a final set of unseen observations called the *test set*. This set needs to be separated from the one used for the CV procedure in order to guarantee that none of the examples it contains have been seen by the model.

An illustration of the decomposition of the full set of observations S is provided on Figure 3.3

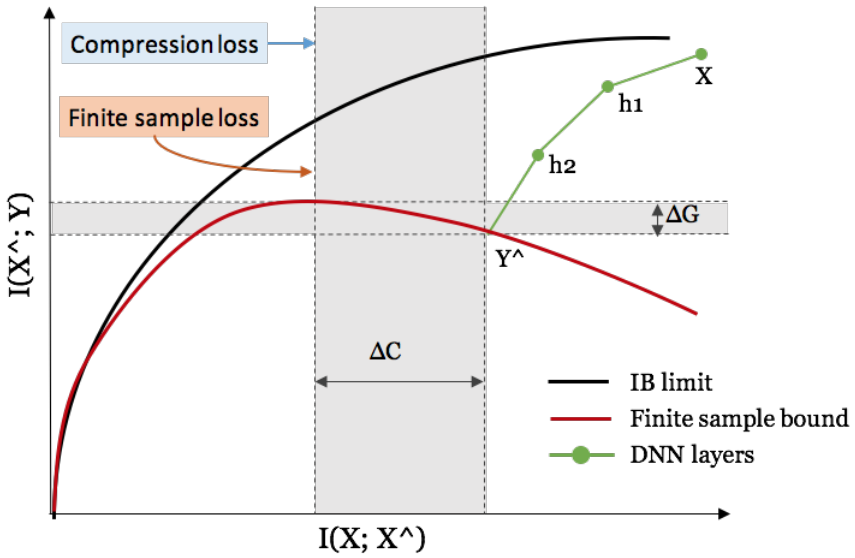


Figure 3.4: *A qualitative information plane, with a hypothesized path of the layers in a typical DNN (green line) on the training data. The black line is the optimal achievable IB limit, and the red line corresponds to the IB curve estimated from a finite set of samples. The performance of the representations are measured with respect to their distance to the most informative point of the red curve.*

3.2 A novel theory of learning ?

It was proposed in [30] that supervised learning could be formulated in the Information Bottleneck framework as a new inventive paradigm to analyze, evaluate, and compare the performances of models with the common currency that is mutual information. More specifically, the authors suggested that the goal of supervised learning is to find representations that are IB optimal, and that the quality of a model should be evaluated by its distance to the IB curve in the information plane, a key element for any IB analysis. In this view, the accuracy of the model is measured by $I(Y;\hat{Y})$, where $f_{\boldsymbol{\theta}}(X) = \hat{Y}$ denotes the prediction of the model, and its complexity is measured by $I(\hat{Y};X)$. Figure 3.4[2] provides a hypothetical IB analysis of the performance of a supervised learning model.

A preliminary exploration of this idea was conducted in [30] to showcase how deep neural networks (DNNs) can be studied in the IB framework. One of their biggest claims in this paper is that the success of Deep Learning can be accounted for by the fact that they implicitly try to solve the IB problem given in Equation 2.9, a claim reiterated in a follow-up work [31] providing empirical evidence to their

[2]Source of the illustration : https://lilianweng.github.io/lil-log/2017/09/28/anatomize-deep-learning-with-information-theory.html

affirmations and offering new insights regarding the learning dynamics of DNNs, the entirety of which is now referred to as the IB theory of Deep Learning.

Since the publication of these two papers, most of the research related to the link between the IB and supervised learning has been focused on either studying the validity of the claims made in [31] on Deep Learning and furthering the theory [32–34, 34–40], or developing new potentially promising DNNs architectures based on the IB [7, 11–14, 16, 41–43].

Concurrently, others have provided theoretical justifications for the usage of the IB functional as a learning objective for supervised methods by providing new tight bounds on mutual information [44], and by proving that it is a useful measure of generalization in restricted settings [6, 45, 46]. Moreover, it has been shown that minimizing $I(Y;\hat{Y})$ is equivalent to minimizing the cross-entropy, a common objective function used for classification tasks [47].

Conversely, several shortcomings of the IB framework have been identified. First, [33] noted that the IB problem can be ill-posed for certain deterministic tasks, that is when there exists an exact mapping between the input and the output $f(X) = Y$, where the relevant information $I(X;Y)$ happens to be unbounded. This was proven to be the case when X and Y are continuous-valued, or if there is at least one continuous component in their respective distribution, given that $H(Y)$ is finite. Moreover, both [32] and [33] have demonstrated that unbounded mutual information can also occur when evaluating models that learn deterministic function $f_{\boldsymbol{\theta}}$. In such cases, they showed that for continuous representations $\hat{Y}$ with finite entropy, $I(\hat{Y};X)$ must be infinite, rendering the evaluation of the complexity irrelevant for such models. A second issue noted by [31] is that the quantity $I(\hat{Y};X)$ does not necessarily correlate with the complexity of the class of function it comes from, nor with the computational complexity of the model, questioning its usage as a measure of complexity. Besides, it is not clear if $I(X;\hat{Y})$ is generally a useful penalizer as empirical evidence in [32] indicates that low values of $I(\hat{Y};X)$ do not relate causally to improved generalization performances. Finally, doubts have been cast on whether or not IB optimal solutions all correspond to desirable solutions [28, 32].

Research towards a better understanding of the IB in the context of supervised learning is still ongoing, and the debate on the relevance of the IB framework to evaluate and compare model performances is not closed. A controversial talking point has been on the techniques used to estimate mutual information, as some researchers have criticized the interpretations made on the dynamics observed in the information plane [32, 48–50].

In the next section, we present new results regarding the capabilities of the IB framework to assess different aspects of supervised learning.

3.3 New insights in classification tasks

Our analysis will be concerned with classification tasks. Many, if not most, of real-world classification tasks are deterministic in nature. Others might not be truly deterministic, but in practice it is often the case that the set of unique inputs in a datasets are assigned to only one class. While not all classifications tasks are deterministic, we shall assume in the following that the set of observations S at our disposal defines one. Moreover, we shall also assume that the representations considered do not lead to unbounded mutual information. This can be achieved for instance by models that build discrete representations, or that learn stochastic maps $f_{\boldsymbol{\theta}}$.

Recall that for deterministic tasks, the shape of the IB curve is piece-wise linear with the elbow exactly located on the point $(H(Y), H(Y))$ in the information plane. From now on, we shall consider that it the only IB optimal solution since it is clear that all other points of the IB curve are suboptimal; either too complex, or not relevant enough. Furthermore, it is interesting to observe that all optimal IB representations must be located somewhere on the line $I_Y = H(Y)$. For this reason, the goal of the IB in deterministic settings can be stated as finding a representation $\hat{Y}$ such that $I(Y, \hat{Y})$ and $I(\hat{Y}, X)$ are as close as possible to $H(Y)$.

In the following, we shall study two aspects of supervised learning models in the bottleneck framework :

1. The fit to the data
2. The generalization performance.

Regarding the first aspect, we show that while the IB optimal solution characterizes models that fit perfectly to the training data, it also consists of representations that common loss functions judge as poor fit. For the second one, we will demonstrate the same caveat but for the generalization performances of a model.

Evaluating the fit to the data

Let IB_{train} be the curve estimated from the training data. By IB standards, the goodness of the fit of a model to the training set is to be measured by how distant it is to IB_{train} optimal point in the information plane.

In the case of a model which achieves a training error of zero as measured by typical loss function, it must also be an IB optimal solution since $Y = f_{\boldsymbol{\theta}}(X)$.

Now consider a model such that $\bar{\text{Err}} \neq 0$ with standard loss function. In this case, the IB judgment of the fit will depend on whether or not there exists an invertible transformation such that $\sigma(\hat{Y}) = Y$. If such transformation exists, then by invariance of the mutual information to this type of mapping, $I(Y; \hat{Y}) =$

$I(Y;\sigma(\hat{Y})) = H(Y)$, and so the representation is IB optimal, even though it is not by traditional accuracy measures. An illustrative example of such a setting is described on Table 3.1. This model is only able to correctly classify 25% of the

Input	Prediction	Output
$\boldsymbol{x}^{(1)}$	y_1	y_2
$\boldsymbol{x}^{(2)}$	y_1	y_2
$\boldsymbol{x}^{(3)}$	y_3	y_3
$\boldsymbol{x}^{(4)}$	y_2	y_1

Table 3.1: *A set of 4 samples belonging to one of 3 possible classes. The model is only able to correctly predict the class of $\boldsymbol{x}^{(3)}$, and mismatches the labels of the others.*

samples in the dataset, but since permutation transformations are invertible, the IB assesses that it is optimal.

Consequently, the IB optimal solution is not a sufficient measure of a model's fit to the training data.

Evaluating the generalization performances

As described in 3.1, the generalization performances of a model are assessed by its accuracy on a test set. In the bottleneck framework, it is proposed to assess it by its distance to the optimal solution on IB_{full}, i.e. the curve estimated empirically from the set of full observations, as it should be the best estimate achievable of the true optimal solution.

Our analysis is concerned with the impact of the training set on the IB evaluation of the generalization capabilities of a model. We consider two situations. In the first, the training set is not representative of the full set of observations, especially with respect to the empirical distribution of the output. In the second, we consider that the training set does represent well the full set.

In the first case, the IB problem defined by S and S_{train} are basically different, and as such their solutions are not the same. In particular, if the distribution of the outputs in S_{train} is more homogeneous than the one in S, then IB_{train} will be located above IB_{full}. This is because the entropy of the output is higher in S_{train} than in S. Conversely, IB_{train} will be located under IB_{full} if the output distribution is less evenly distributed in S_{train} than in S. In any case, if the model perfectly fit the training data, then it will reach the IB optimal point of IB_{train}. Hence, in the former case, the representation it learns is outside the feasible region defined by IB_{full}. However, since the two IB problems are different, there is no contradiction. Overall, it seems that for this type of scenario the distance to the IB optimal solution is an irrelevant measure of generalization as the two IB problem are non-identical.

Now assuming that the training set and the full set have similar empirical distributions for the output variable, and that there exists a model which perfectly fits the training data. In that case, its representation should be IB_{train} optimal but also IB_{full} optimal since the two curves coincides. So, by the IB standards its generalization capabilities are perfect. However, as explained in Section 3.1, perfect prediction performances on the training set often implies larger test error, implying that this kind of model do not in fact generalize well. However, it is also true that a model which perfectly classify all samples in S must be IB_{full} optimal, demonstrating that the IB optimal representation is not a sufficient measure of the generalization capacities of a model.

3.4 Summary and discussion

In summary :

- Given a labeled set of examples, supervised learning methods aim to learn a map $f_{\boldsymbol{\theta}}$ characterized by a set of tuning parameters $\boldsymbol{\theta}$ which is able to predict efficiently the outcome y of any input $\boldsymbol{x}$.
- In general, models are easily able to predict the outcome of examples which they have already seen. However, they are evaluated by their generalization performances, that is their capacity to predict correctly the outcome of unseen inputs, for which they often have more difficulties.
- Model which fits too much the training data to the cost of loosing accuracy on sets of unseen examples are said to overfit. Decreasing overfitting can be achieved by limiting the complexity of the model.
- The IB theory of learning proposes that models performances should be evaluated with respect to solutions to the IB problem, which are assumed to be the best possible representations achievable.
- As of now, no consensus has been reached on the usefulness of this theory to understand supervised learning, although great efforts have been made in this regard.
- For deterministic classification tasks, the IB optimal solution is neither sufficient to assess a model fit to the training data, nor to evaluate its generalizations performances.

In the Chapter 5, it will be shown that our theoretical analysis of Section 3.3 is corroborated by empirical results. In addition, our experiments also demonstrate

that there exist representations not considered to be optimal in the IB framework, but with better generalizations performances than optimal IB_{full} solutions. Our model of choice to carry out these experiments is the well known decision tree method, which will be introduced in details in the following chapter.

Chapter 4

Decision trees

Before presenting our experiments, we must first introduce the supervised learning model used to carry them out : decision trees. Hence, the goal of this chapter will be to cover all essential part of this method.

First, the idea behind the method as well as its tree structure will be introduced in Section 4.1. Then, Section 4.2 will give a thorough explanation of the CART algorithm, an established method to construct decision trees. After that, a review of common methods used to limit the complexity of the model will be presented in Section 4.3. Finally, Section 4.4 will cover the main advantages and limitations of tree models, and the chapter will end with a summary of the principal takeaways given in Section 4.5.

The references used throughout this chapter are the chapter 3 of [5], section 9.2 of [4], and most of [51].

4.1 Decision tree representation

A typical representation of a decision tree model is presented on Figure 4.1[1]. The model predicts whether a passenger from the Titanic was likely to survive or not based on three features, namely gender, age and sibsp (number of spouses or children aboard). The root node tests the gender feature, and branches into two children nodes. The left one which corresponds to the observation that the passenger is male, and the right one which corresponds to the observation that it is female. The right node is not split, because the model assesses that it has enough information to make a good prediction on the chances of survival for female passengers. So, it makes a prediction, in this case that female passengers were likely to survive the sinking of the ship. On the other hand, the child on the left is

[1]Source of the illustration : https://commons.wikimedia.org/wiki/File:Decision_Tree.jpg

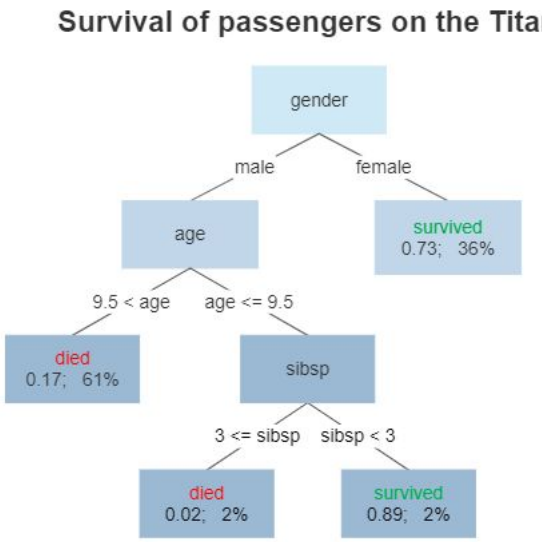


Figure 4.1: *Typical tree representation of a decision tree model illustrating the rules it has derived to make its prediction, here being whether a passenger of the Titanic likely survived or not.*

split, meaning that the model believes it needs more information than knowing that the passenger is a male to provide a good prediction. In the end, the model predicts that the chances of survival were good if the passenger was (i) a female or (ii) a male younger than 9.5 years with no more than than 3 siblings.

In any such representation of decision trees model, *internal nodes* correspond to nodes that are split, and *terminal nodes* to nodes that are not. An internal node tests a feature of the input, and branches into the possible values that this feature can take, whereas a terminal node is assigned a target value to which it maps inputs that fall in it. Basically, a tree consists of a set of decision rules which correspond to all the paths in the tree that can be taken from the root to any terminal nodes. A sample fed to the model will follow the path to which the values of its features correspond until it reaches a terminal node where it is mapped to a target value. If the target value is discrete, the model is called a *classification tree*, and a *regression tree* if it is continuous.

Decision trees are constructed in two steps. First, they build finer and finer partitions of the feature space $\mathcal{X}$ in which the regions are maximally informative about the target value. In the tree representation, the nodes at a given depth represent such a partition. Then, once it judges that the partition is good enough, it assigns a label to each region, and so all samples inside it are mapped to the same

target value. The regions of the final partition correspond to the terminal nodes of the tree structure. For classification trees, the predicted label of a region correspond to the dominant class present in the observations inside it. In general, given a partition $R_1, R_2, \ldots, R_M$, the predicted target value is modeled as a constant $\hat{y}_m$ in each region R_m. It is given by the prediction map,

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} \hat{y}_m I(\boldsymbol{x} \in R_m) \tag{4.1}$$

which takes instance $\boldsymbol{x} = (x_1, x_2, \ldots, x_p) \in \mathcal{X}$ as input, and depending on the region to which they belong, maps them to the associated target value. Another example of a decision tree model is shown on Figure 4.2, where the partition built can be seen on the top-right panel, and the bottom-right panel showcases the target values associated to each regions. The finer the partition is, the more regions it contains and so the deeper the tree.

4.2 Algorithm

The two most popular algorithms to build decision trees are C4.5 and CART. They are similar in many aspects, but a key difference is that CART splits nodes into just two children, whereas C4.5 allows for multiway splits if needed. This does not restrict the partition capabilities of CART since multiway splits can be achieved by a series of binary splits as depicted on Figure 4.3. One reason binary splitting is preferred, is that multiway splits fragment the data too rapidly, leaving too few samples down the tree.

In the following, we will focus on the CART algorithm, whose name is an acronym for Classification And Regression Tree, which is capable of builduig both classification and regression trees, as its name suggests. With the exception of the binary splitting, most of the techniques used in CART are the same used in C4.5, so the following explanation also apply to C4.5.

CART works as follows. Given a set of N observations,

$$S = \{(\boldsymbol{x^{(i)}}, y^{(i)}) = (x_1^{(i)}, x_2^{(i)}, \ldots, x_p^{(i)}, y^{(i)})\}_{i=1}^{N}$$

made up of vectors $\boldsymbol{x^{(i)}}$ with p features and their observed responses $y^{(i)}$ (a label in a classification setting), CART builds a partition of the feature space such that the samples with similar response values are grouped together. To do so, it uses a recursive procedure that begins with the full set S. CART then chooses a feature j and one of its value s that best splits the set of samples in two disjoint subsets. One with the samples that satisfy $x_j^{(i)} \leq s$, and the other with the samples that satisfy $x_j^{(i)} > s$. Essentially, dividing the feature space in two. This procedure

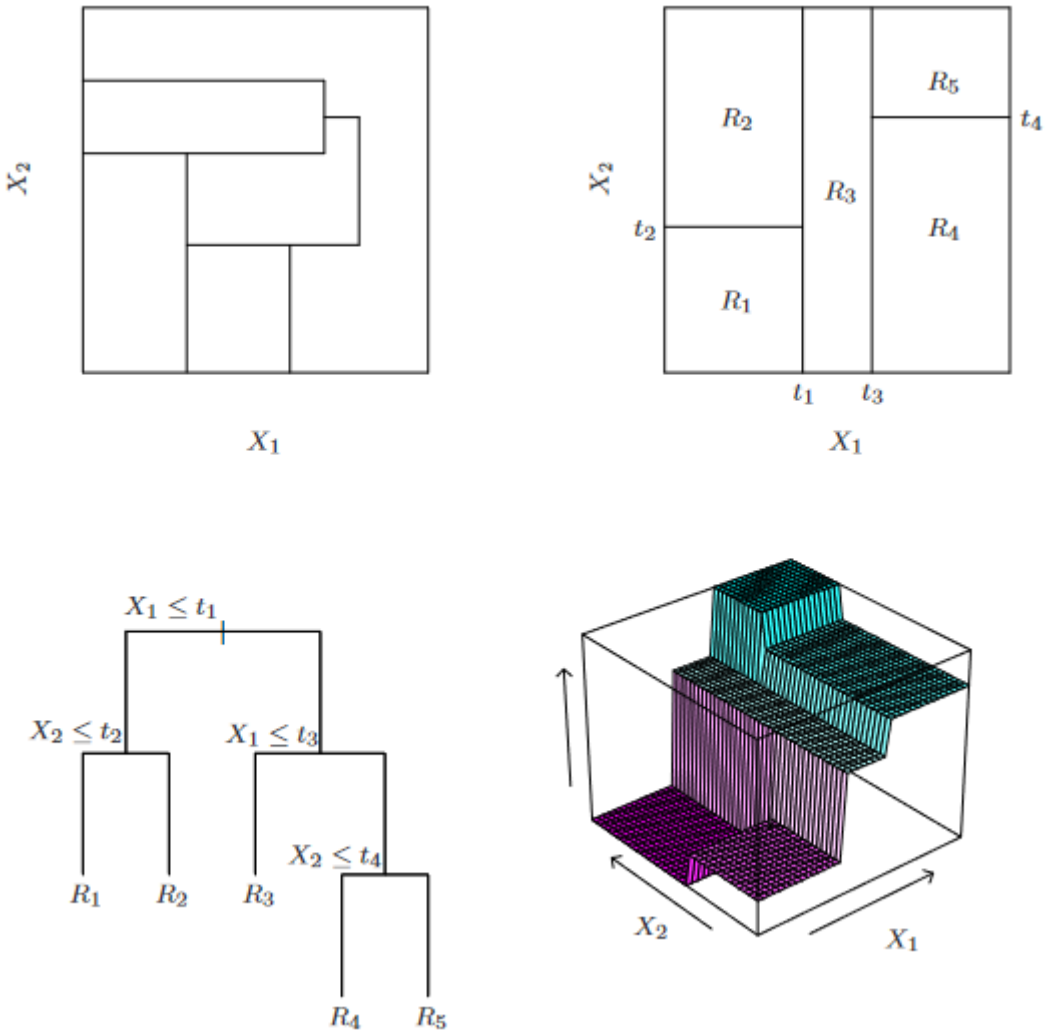


Figure 4.2: *Different representation of a CART model. Top left panel shows a partition unobtainable from recursive binary splitting. Top right panel illustrates a partition achievable with CART. Bottom left panel shows the tree corresponding to the partition in the top right panel, and a perspective plot of the prediction surface appears in the bottom right panel [4].*

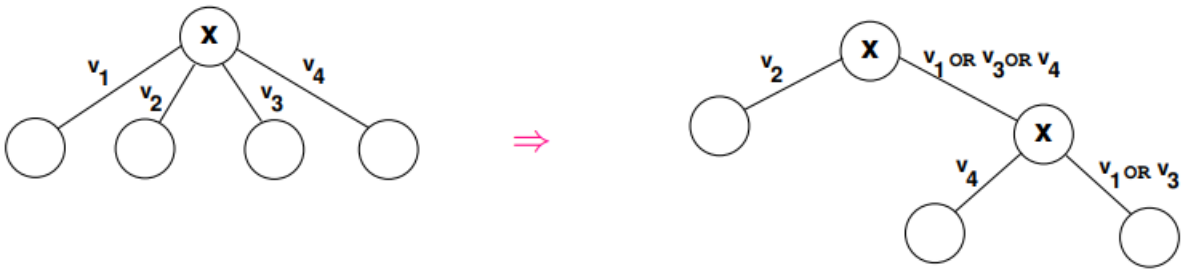


Figure 4.3: *A multiway split can always be achieved by a succession of binary splits [5].*

is then applied to the two subsets, launching the recursion. If a subset contains instances with exclusively the same target variable values or if splitting no longer adds value to the prediction, then it does not split this subset. Typically, metrics used to evaluate the quality of a split point (j, s) measure the homogeneity of the target variable within the two subsets, as we shall now explain in more detail.

In a region R_m of the feature space, let $S_m \subset S$ denote the subset of data that belongs to it. Consider the split (j, s) which partitions the region R_m into two new regions $R_m^{\text{left}}(j, s)$ and $R_m^{\text{right}}(j, s)$. The subsets of samples inside each are denoted respectively by $S_m^{\text{left}}(j, s)$ and $S_m^{\text{right}}(j, s)$. The quality of the split is evaluated using an impurity measure Q which is used to define its cost given by,

$$C(S_m, j, s) = \frac{N_m^{\text{left}}}{N_m} Q(S_m^{\text{left}}(j, s)) + \frac{N_m^{\text{right}}}{N_m} Q(S_m^{\text{right}}(j, s))$$

with $N_m^{\text{left}}, N_m^{\text{right}}$ the number of samples in $S_m^{\text{left}}(j, s)$ and $S_m^{\text{right}}(j, s)$ respectively. After evaluating several splits, CART chooses the one that minimizes the cost function. In general, a good split either divides the training set in two equal parts $(N_m^{\text{left}} \approx N_m^{\text{right}})$, or achieves low impurity function value, or even does both.

CART is an example of a greedy algorithms which chooses the best split point at each step, and does not concern itself with finding a globally optimal tree as it is too computationally expensive. Hence, CART grows the tree only as large as needed in order to classify the available samples in S.

There exists several types of impurity measure Q for regression and classification trees. Here, we present the most commonly used.

Classification trees

For a classification task with classes $k = 1, \ldots, K$, the proportion of observations of class k in the subset S_m is given by,

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{(\bar{x}^{(i)}, y^{(i)}) \in S_m} I(y^{(i)} = k). \tag{4.2}$$

The cost functions used in classification trees usually measure the impurity of a region by how mixed the classes of its observations are. A subset is said to be pure if it contains only instances from a single class, and it is said to be impure if the classes have the same proportions. The three most common impurity measure used in classification trees are :

- Gini impurity : $Gini(S_m) = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) = 1 - \sum_{k=1}^{K} \hat{p}_{mk}^2$
- Entropy in the classes : $H(S_m) = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$
- Misclassification : $Mis(S_m) = 1 - \max_k \hat{p}_{mk}$

These measures are compared on Figure 4.4 for a random variable with only two values. We see that entropy and Gini are more sensitive to changes to the distribution of Y than the misclassification rate. That is why, if the training set S is slightly modified, the resulting tree might end up being very different.

In some implementations of CART, it is preferred to compute the *gain* instead of the cost of a split. The gain of a split is calculated as ,

$$G(S_m, j, s) = Q(S_m) - \frac{N_m^{\text{left}}}{N_m} Q(S_m^{\text{left}}(j, s)) - \frac{N_m^{\text{right}}}{N_m} Q(S_m^{\text{right}}(j, s))$$

and the best split is the one that maximizes the gain. Since the first term is a constant in the optimization process, the minimization of the cost is equivalent to the maximization of the gain. When the entropy is used, the gain function is called the *information gain*. It corresponds to the information provided about the target value given the value taken by the feature. In fact, it corresponds to the mutual information between the output and the feature x_j :

$$\begin{aligned} IG(S_m, j, s) &= H(S_m) - H(S_m | x_j = s) \\ &= H(S_m) - \frac{N_m^{\text{left}}}{N_m} H(S_m^{\text{left}}(j, s)) - \frac{N_m^{\text{right}}}{N_m} H(S_m^{\text{right}}(j, s)) \end{aligned} \tag{4.3}$$

In classification trees, the predicted target value in a leaf node m in Equation 4.1 is given by the majority class in this node

$$\hat{y}_m = \arg\max_k \hat{p}_{mk}$$

Regression trees

The most common measure used in regression trees is the mean squared error

$$MSE(S_m) = \frac{1}{N_m} \sum_{(\boldsymbol{x}^{(i)}, y^{(i)}) \in S_m} (y^{(i)} - \bar{y}_m)^2$$

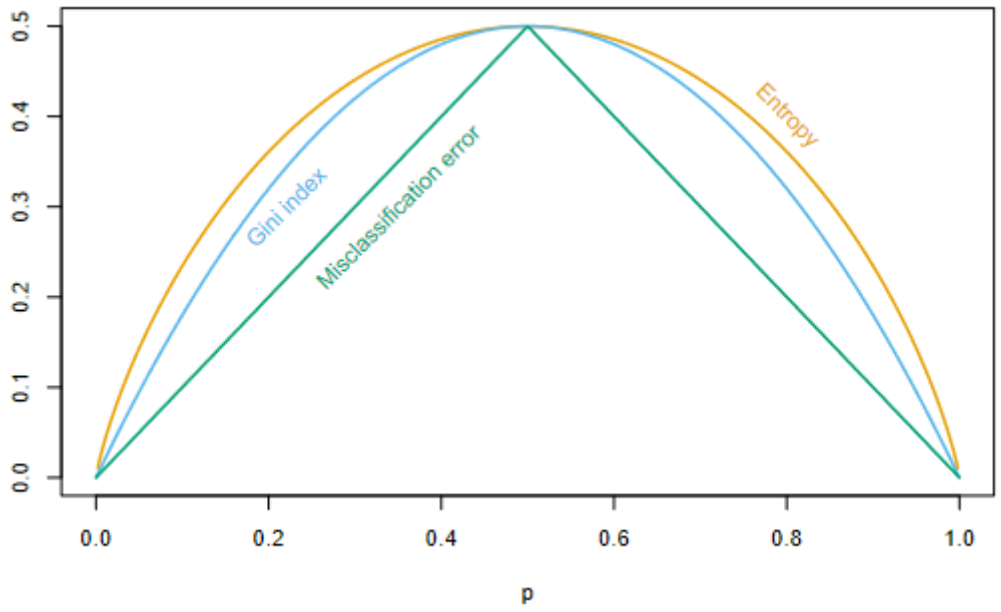


Figure 4.4: *Node impurity measures as a function of the proportion p of observations in one of the two possible classes [4].*

with the predicted value $\hat{y}_m$ in terminal nodes set as the average of the targets value present in the node

$$\bar{y}_m = \frac{1}{N_m} \sum_{(\boldsymbol{x}^{(i)}, y^{(i)}) \in S_m} y^{(i)}.$$

Other metrics exists such as the mean absolute error and the half Poisson deviance.

4.3 Pruning

The size of a tree, here defined by its number of terminal nodes, is a measure of the model complexity. From Section 3.1, we know that if the model is too simple, its prediction abilities will be limited. But if it is too complex, it might fit the noise in the training set and overfit. In order, to reduce the chances of overfitting, techniques have been proposed to limit the size of the tree. The simplest consists of directly restricting the depth of the tree. More sophisticated approaches to control the size of the tree involves removing wisely the nodes that do not provide enough gain to the prediction capabilities of the model. The two main approaches are called *pre-pruning* and *post-pruning*, which we now presents.

Pre-pruning

A first strategy is to only split nodes that bring enough gains. There are two usual methods to perform this :

(i) Only split nodes with a certain number of instances.

(ii) Only split nodes which have a level of impurity above a given threshold value.

An inconvenient to this strategy is that local decisions might not be globally optimal. What may seem like a pointless split may in reality lead to a valuable split below it. Hence, pre-pruning can sometimes lead to worse generalization performances. On the contrary, the post-pruning strategy always outputs a model with better generalization performances, for this reason it is often the preferred method to reduce the model complexity.

Post-pruning

Post-pruning works in two phases. Initially, the tree is allowed to grow freely as large as needed to correctly classify all training samples. Then in a second phase, the tree is pruned using the *minimal cost-complexity pruning* method. This procedure works as follows. Let $\mathcal{T}_0$ represent our initial tree and let $\mathcal{T} \subset \mathcal{T}_0$ be a subtree obtained by pruning nodes from $\mathcal{T}_0$. The algorithm finds the subtree $\mathcal{T}$ that minimizes the cost-complexity measure given by,

$$C_\alpha(\mathcal{T}) = Q(\mathcal{T}) + \alpha|\mathcal{T}| \tag{4.4}$$

where,

- $|\mathcal{T}|$ denotes the size of the tree as measured by its number of terminal nodes,
- $Q(\mathcal{T})$ is either an impurity function (often chosen as the misclassification) or a loss function depending on if it is a classification or a regression tree,
- $\alpha \geq 0$ is called the *complexity parameter*. It controls the trade-off between the complexity of the model and the quality of the fit. Larger values of α equate to a bigger penalty on the size of the tree, resulting in smaller trees. Conversely, smaller values favour larger trees and in particular $\alpha = 0$ simply yields the non-pruned tree $\mathcal{T}_0$.

For each α value, it can be shown that there exists a unique smallest subtree $\mathcal{T}_\alpha$ that minimizes $C_\alpha(\mathcal{T})$. One can find this subtree by evaluating the importance

of each internal node denoted by t, and prune the ones that performs the worst. The evaluation of a node t is done using the *effective* α :

$$\alpha_{eff}(t) = \frac{Q(t) - Q(\mathcal{T}_t)}{|\mathcal{T}_t| - 1} \tag{4.5}$$

where $\mathcal{T}_t$ is defined as the subtree with root node t. The numerator measures how close the purity of the node is to the purity of the terminal nodes of its subtree. Basically, it tries to find out if the subtree $\mathcal{T}_t$ is really useful or not. If it is not, then α_{eff} value will be lower and conversely if it is really useful. The denominator is a penalty on the size of the subtree. Even if the subtree is quiet useful, if it is tremendously large then it will still lead to a small α_{eff}. Overall, small subtrees with low impurity will have the largest effective α.

In summary, the procedure prunes nodes with the lowest α_{eff} - the weakest links - and stops once the pruned tree minimal α_{eff} is larger than a specified value $\hat{\alpha}$.

4.4 Properties

The CART method has a number of properties. In this section, we review some of the most important, which are shared for the majority with other popular decision tree algorithms, notably C4.5 and its successor C5.0.

Advantages

- White-box model : By construction, the CART decision tree model provides the rules used for the predictions it makes. These rules are especially well suited for human understanding as they correspond to simple Boolean logic which follows human cognition more closely than other approaches. Moreover, the tree visualization of the model summarizes these rules in a structured manner.

  However, this benefit is lost in two scenarios : (i) if the features are non-intuitive, e.g. pixels of an image. Then, even with the knowledge of the rules, it is hard to make sense of the decision taken by the model. (ii) If the tree is especially deep, then the rules become overly complex, and again it becomes hard to understand the final decision.

- Versatile : Decision trees are able to handle many types of data. Moreover, they require relatively little data preparation, whereas other supervised learning methods might need some form of data pre-processing. Examples include the need for normalized variables or the creation of dummy variables

to account for discrete features. Some implementations of CART even include in-build techniques to treat missing attribute values.

- Scalable : Decision trees implementations are often highly efficient in comparison to other popular machine learning techniques. Models can be trained on large amounts of data with standard computing means in a sensible amount of time.

- In-build feature selection : Decision trees build a hierarchy of the features ranked by their importance. The features deemed the most important are the first selected by the model and are closer to the root of the tree. Whereas, lesser influential variables are either deeper in the tree or are not selected at all.

Limitations

- Greedy effect : CART uses a greedy strategy and so does not guarantee that the resulting tree is globally optimal. Smaller trees which fit as well on the training data might exist.

- Overfitting : Decision trees, as most supervised learning methods, are prone to overfitting. If no restrictions are put on the size of the tree, then it will grow as large as it needs to be to correctly classify all training samples. This typically leads to over-complex trees that have poor generalization performances.

  Solutions to reduce overfitting often take the form of pruning mechanisms that limits the size of the tree, as presented in the previous section.

- Instability : Tree models have a large variance. Minor changes in the training set can result in a wholly different tree structure.

  A well established solution to alleviate this issue is to average many trees to reduce the variance of the model (*bagging*).

4.5 Summary

In this chapter, we presented the well known machine learning method known as decision tree, with a particular attention given to the CART algorithm. All essential characteristics of the method were covered from the intuition behind the model to the details of its inner workings.

In summary, the main takeaways are :

- Decision trees build a partition of the feature space in which regions are as pure as possible. The predicted response is chosen either as the dominant class for classification trees or the average value for regression trees. Deeper trees corresponds to finer partitions.
- The model can be represented as a tree in which nodes represent a feature that tests the values of the samples. Out-going branches of a node designate the value taken by its assigned feature - the decision.
- The CART algorithm is a recursive procedure that builds binary decision trees by splitting the feature space into several regions, choosing the best splits as the feature and one of its value which minimizes a cost function.
- There are three main methods to restrict the size of the tree in order to limit overfitting : (i) directly limit the depth to a certain value, (ii) only split nodes which bring enough gain, or (iii) grow the tree as large as needed to perfectly predict all examples, then prune ineffective nodes.
- Decision trees are white-box models. Moreover, they are able to handle many types of data and scale well to larger datasets. However, they are prone to overfitting and small changes in the training data can results in vastly different trees. Plus, they use a greedy procedure to build the model, and therefore do not guarantee that the tree is globally optimal.

In the following chapter, the first to our knowledge IB analysis of decision tree will be presented. It will be explained how decision trees can be interpreted in the IB framework, and use these considerations to perform several experiments which will be presented at the end.

Chapter 5

Empirical analysis and decision trees representations

This chapter is divided into two parts. The first will be dedicated to the interpretation of decision trees in the IB framework. It will be shown that the successive partitions built by the model at each depth level can be seen as different intermediate representations, which shall be referred to as its layer representations. In the second part, the set up and results of several experiments studying the claims laid in Section 3.3 will be presented.

Decision trees have been chosen for our experiments for several reasons. First, their intermediate representations are discrete-valued, and so there is no risk of unbounded mutual information. Secondly, the quality of the partitions/representations can be easily visualized for small feature space, thus making it easy to assess the quality of the model with respect to the fit to the data, and to assess its generalization performances. Finally, the fact that they possess representations of varying complexity is particularly interesting for an analysis in the information plane.

The structure of this chapter is as follows. Section 5.1 will explain how decision trees fit in the bottleneck framework. In Section 5.2, expressions for the information content of the layers representations will be derived explicitly. The theoretical analysis of Chapter 3 will be applied to decision trees in Section 5.3. Finally, Section 5.4 will provide experimental verification of the expectations that have been laid in Chapter 3, plus it will also be shown that there exist representations that are not IB optimal but which generalizes better than solutions that belong to the IB curve.

5.1 Decisions trees layers representation

A natural definition for the intermediate representations of a decision tree is the partition of the feature space $\mathcal{X}$ built at each depth level. So, in the tree representation of the model, the layer ℓ is the set of M_ℓ nodes and leaves at depth ℓ, plus the leaves of the previous layers. This set of regions/nodes is denoted by,

$$\{R_1^{(\ell)}, \ldots, R_{M_l}^{(\ell)}\} \tag{5.1}$$

In order for the layers of the tree to fit in the IB framework, they are interpreted as random variables taking values in the set of regions Equation 5.1. So, a layer ℓ is defined as the following representation

$$T_l = \begin{cases} R_1^{(\ell)}, & p(R_1^{(\ell)}) \\ \vdots & \vdots \\ R_{M_\ell}^{(\ell)}, & p(R_{M_\ell}^{(\ell)}) \end{cases} \tag{5.2}$$

where the probability of the event $R_m^{(l)}$ is given by the probability that the a realization of X belongs to this region. These probabilities can be computed empirically from the set of observations S using the following ratio,

$$p(R_m^{(\ell)}) = \frac{\#\text{samples in } R_m^{(\ell)}}{\#\text{samples}}. \tag{5.3}$$

Each layer representation T_ℓ satisfies the following Markov chain :

$$T_\ell - X - Y. \tag{5.4}$$

Furthermore, the representations are linked together by another Markov chain, which is given by

$$T_1 - T_2 - \ldots - T_{L-1} - T_L - X - Y. \tag{5.5}$$

Consequently, they satisfy the data processing inequalities

$$I(T_L; X) \geq I(T_{L-1}; X) \geq \ldots \geq I(T_2; X) \geq I(T_1; X), \tag{5.6}$$

and,

$$I(Y; X) \geq I(Y; T_L) \geq I(Y; T_{L-1}) \geq \ldots \geq I(Y; T_2) \geq I(Y; T_1). \tag{5.7}$$

The first signifies that deeper layers are more informative about X, which is not surprising since they incorporate more information about the feature space than layers closer to the root. The second one is implied by the first, since the information about Y can solely be obtained through X.

Prediction representation

Until now, we have only considered the partitions corresponding to each depth level of the tree, and gave each one a representation that fits in the IB. However, to fully capture the method, we still need to attach a representation to the map in Equation 4.1 used by the model to make predictions. Adapting slightly our notations to take into account the previous considerations, the map is given by,

$$f(\boldsymbol{x}) = \sum_{m=1}^{M_L} \hat{y}_m I(\boldsymbol{x} \in R_m^{(L)})$$

and assigns a value $\hat{y}_m$ to each regions $R_m^{(L)}$ in the partition of the last layer L.

We denote by $\hat{Y}$ the representation for this map, and name it the *prediction representation*. Like the others, it is defined as a random variable but with values in the support set $\mathcal{Y}$ of the target variable Y. For instance, if Y is discrete-valued with K classes, then the representation $\hat{Y}$ is characterized by the probability distribution,

$$\hat{Y} = \begin{cases} \hat{y}_1, & p(\hat{y}_1) \\ \vdots & \vdots \\ \hat{y}_K, & p(\hat{y}_K) \end{cases}$$

with probabilities given by the following ratio,

$$p(\hat{y}_k) = \frac{\#\text{samples mapped to } \hat{y}_k}{\#\text{samples}}.$$

Otherwise, if Y is continuous, then the model will output at most M_L different values, and so the above expression of $\hat{Y}$ is simply adjusted to take into account the number of output values. The expression for the probabilities stays unchanged.

Finally, the prediction representation satisfies the Markov chain $\hat{Y} - T_L - X - Y$ since it is conditionally independent of both X and Y given T_L. Hence, it must satisfies the following data-processing inequality,

$$I(\hat{Y}; T_L) \geq I(\hat{Y}; X) \geq I(\hat{Y}; Y)$$

meaning that the prediction $\hat{Y}$ only possesses information about X and Y through the last layer representation T_L.

5.2 Layers information content

From the data processing inequalities Equation 5.6 and 5.7, it is already clear that the layers, ordered by $\ell = 1, \ldots, L$, should follow an increasing pattern in the information plane. Now to further our analysis, but also because it will be useful for our experiments, we derive explicit expressions for $H(T_\ell)$, $I(Y; T_\ell), I(T_\ell; X)$, and for $H(\hat{Y}), I(Y; \hat{Y}), I(\hat{Y}; X)$.

Entropy of the representations

By definition of the entropy, Equation 2.1, we have the following expression for the entropy of a layer T_ℓ,

$$H(T_\ell) = -\sum_{m=1}^{M_\ell} p(R_m^{(\ell)}) \log p(R_m^{(\ell)}).$$

where the probabilities can be computed empirically from the set of examples S using the ratio in Equation 5.3.

For the prediction representation, if there are K different output values then its entropy is given by,

$$H(\hat{Y}) = -\sum_{k=1}^{K} p(\hat{y}_k) \log p(\hat{y}_k).$$

Accuracy measure I_Y

In the IB, the accuracy of a layer is measured with $I_Y = I(Y;T_\ell)$. It can be expressed in terms of the entropy $H(T_\ell)$ and the conditional entropy $H(T_\ell|Y)$ by the relation,

$$I(Y;T_\ell) = H(T_\ell) - H(T_\ell|Y).$$

Hence, we only need to compute the conditional entropy term. Depending on whether Y is discrete or continuous, the conditional entropy is given respectively by Equation 2.3 and 2.5.

- For discrete Y:

$$\begin{aligned} H(T_\ell|Y) &= \sum_{y\in\mathcal{Y}} p(y) H(T_\ell|y) \\ &= -\sum_{y\in\mathcal{Y}} p(y) \sum_{m=1}^{M_\ell} p(R_m^{(\ell)}|y) \log p(R_m^{(\ell)}|y). \end{aligned}$$

- For continuous Y:

$$\begin{aligned} H(T_\ell|Y) &= \int_{\mathcal{Y}} dy\; p(y) H(T_\ell|y) \\ &= -\int_{\mathcal{Y}} dy\; p(y) \sum_{m=1}^{M_\ell} p(R_m^{(\ell)}|y) \log p(R_m^{(\ell)}|y) \end{aligned}$$

In both cases, the probabilities can be estimated from the set of samples S by,

$$p(y) = \frac{\#\text{samples labeled } y}{\#\text{samples}}$$

and,

$$p(R_M^{(\ell)}|y) = \frac{\#\text{samples in } R_m^{(\ell)} \text{ labeled } y}{\#\text{samples labeled } y}.$$

As for the prediction representation taking K possible values, the conditional entropy is given for discrete and continuous Y by :

- For discrete Y:

$$\begin{aligned} H(\hat{Y}|Y) &= \sum_{y\in\mathcal{Y}} p(y) H(\hat{Y}|y) \\ &= -\sum_{y\in\mathcal{Y}} p(y) \sum_{k=1}^{K} p(\hat{y}_k|y) \log p(\hat{y}_k|y). \end{aligned}$$

- For continuous Y:

$$\begin{aligned} H(\hat{Y}|Y) &= \int_{\mathcal{Y}} dy\ p(y) H(\hat{Y}|y) \\ &= -\int_{\mathcal{Y}} dy\ p(y) \sum_{k=1}^{K} p(\hat{y}_k|y) \log p(\hat{y}_k|y) \end{aligned}$$

where the probabilities are estimated from S with the ratio,

$$p(\hat{y}_k|y) = \frac{\#\text{samples mapped to } \hat{y}_k \text{ and labeled } y}{\#\text{samples labeled } y}.$$

Complexity measure I_X

It turns out that the complexity of a layer ℓ, $I_X = I(T_\ell; X)$, is simply reduced to its entropy. This can be seen using again the mutual information expression in Equation 2.6,

$$I(X; T_\ell) = H(T_\ell) - H(T_\ell|X).$$

Naturally, the conditional entropy term is null because the relation between the input and the representation is deterministic (it is a surjective function which quantizes the input space). So if X is known, then there is no uncertainty on T_ℓ. Therefore the compression measure becomes,

$$I(T_\ell; X) = H(T_\ell).$$

Notice that this expression highlights the increasing pattern of the complexity measure because deeper layers are more informative about X

Since the prediction map is also deterministic, the same considerations holds and so,

$$I(\hat{Y}, X) = H(\hat{Y}). \tag{5.8}$$

In summary, deeper layers consist of more complex representations of the input, and as a result possess higher values of I_X. This increase in complexity can be accompanied by an increase in the accuracy measure I_Y as well if the model makes the right choices of partitioning.

Discussion

Overall, the layer representations can be nicely characterized by their information content. They are discrete valued, and so can be estimated more easily than continuous valued representations. Moreover, this also implies that there is also no risk of unbounded complexity I_X for any layers T_ℓ. Only the prediction representation $\hat{Y}$ can suffer from this issue, since it takes continuous values for regression task. In any case, our experiments will be focused on classification tasks so there is no risk of infinite mutual information. Beforehand, the next section will set up the theoretical behaviours of the layers in the information plane based on the results obtained in Chapter 3.

5.3 Layers behavior in the information plane

Before presenting the experiments conducted with decision trees, this section provides a preliminary analysis of the layers behavior in the information plane based the developments made in Chapter 3. Specifically, it is first shown that deeper layer representations should lie closer to the IB_{train} curve as they are able to capture more relevant information. Next, it will be explained how the cost-complexity pruning procedure can be used to obtain representations that do not belong to the IB_{full} curve, but which generalize better than particular representation considered optimal in the bottleneck framework.

Fit to the data

A decision tree perfectly predicts the target values from the set of observations only if all its terminal nodes are pure. In this scenario, the layer of terminal nodes, T_L, possesses all the relevant information to correctly predict the target variable Y from X. Typically, it also contains excessive information about X, which is then removed by the prediction map to only retain the relevant information $I(Y;X)$. Therefore, the representations T_L and $\hat{Y}$ must belong to the IB_{train} curve because they both capture the relevant information. The former should be located on the horizontal part because it is still too complex, but the prediction representation $\hat{Y}$ has to be located right on the elbow of the curve since the two random variables Y and $\hat{Y}$ are exactly the same. Remark that there can not be another layer than

T_L on the IB_{full} curve because tree algorithms stops splitting nodes once they are pure.

Consider now a tree which makes some wrong predictions on the training set because it is too shallow to have its terminal nodes totally pure. Since a perfect mapping exist, there must be features that can be included in the model to decrease the impurity in the nodes, meaning that the model has not extracted all the relevant information $I(Y;X)$ from the input. Thus neither T_L nor $\hat{Y}$ reach the IB_{full} curve. As in the previous case, the prediction map removes information from T_L that the model deems irrelevant to predict Y, so it must be that $I(\hat{Y};X) \leq I(T_L;X)$. However, it will also removes information about Y as well, and so $I(Y;\hat{Y}) < I(Y;T_L)$. This occurs because the prediction map assigns wrong labels to samples inside terminal nodes, thereby removing some relevant information captured by T_L.

Performances on unseen data

Consider a tree model that makes zero error on the training set. From Chapter 4, we know that if there exists a subtree that generalizes better, then the cost-complexity pruning algorithm will find it. But since the process of pruning nodes usually decreases the performance on the training set, the accuracy as measured by the IB $I(Y;\hat{Y})$ will likely decrease as well. So, even though it generalizes better, the pruned model prediction representation does not belong to the IB_{full} curve. This will be empirically demonstrated in the next section.

5.4 Experiments

This section provides empirical evidence to the theoretical considerations laid in Chapter 3 and in the previous section. Each experiments have been carried out on two different classification task : (i) an artificial toy dataset used to easily visualize the model's partition, and (ii) a real world dataset to check that all behaviours are also observed on more complex data. Here is description of the two :

- **Toy dataset** : It consists of the two-dimensional feature space $\mathcal{X} = [0,1] \times [0,1]$ filled by 22.500 evenly spaced samples, which are either labeled "Black" or "Copper". Figure 5.3a shows the sample space with the color representing the associated labels.

- **Real world dataset** : We used the MNIST handwritten digits database. It is made of 70.000 samples of black and white images of 28×28 pixels depicting numbers from 0 to 9. The task is much tougher than the previous

as it contains a total of 784 features instead of 2, which can take values between 0 and 255.

For the first part of our experiments regarding the fit to the data evaluation, the full set is used to train the model in order to have the most accurate estimation of IB_{train} possible. In the second part, the full sets of observations are divided into a training and a test set of respective sizes 6/7 and 1/7 of the totality of the samples.

In all experiments, the information measures of the layers are computed from the empirical distributions of the samples in each node using the expressions derived in Section 5.2. Other information quantities were estimated using non-parametric estimators provided by the NPEET package[1].

Regarding the tree model, a CART implementation from the Sci-kit learn library was used[2]. Unless specified otherwise, the setup uses the information gain and does not include any restrictions on the depth of the tree. Note that the information gain of a layer T_ℓ is equal to the relevance measure I_Y, and so can be visualized in the information plane on the vertical axis.

5.4.1 Evaluating the fit to the data

The results of the experiments are shown on Figure 5.1 and 5.2. Observe that all layers representations and the prediction representation belong to the feasible region, and that the increasing pattern of the information content imposed by the data-processing inequalities, Equation 5.6 and 5.7, is respected by all models. In addition, it can be seen that the first layers often increases the complexity I_X by 1 bit. This signifies that the split divides the set of observations in approximately equal sized parts, and so there is not a big difference in the probability to be in one or the other. For instance, the two regions of the layer T_1 for the toy dataset have the associated probabilities of 0.433 and 0.567. Basically, this confirms that dividing the set in equal parts is a good strategy to maximize the information gain (see Section 4.2). Note that more layers display an increase of 1 bit for MNIST than for toy. It is believed that it is simply because MNIST is more high-dimensional and so possesses a larger number of split points, from which a great number achieve a great gain and split the totality of examples in same sized subsets.

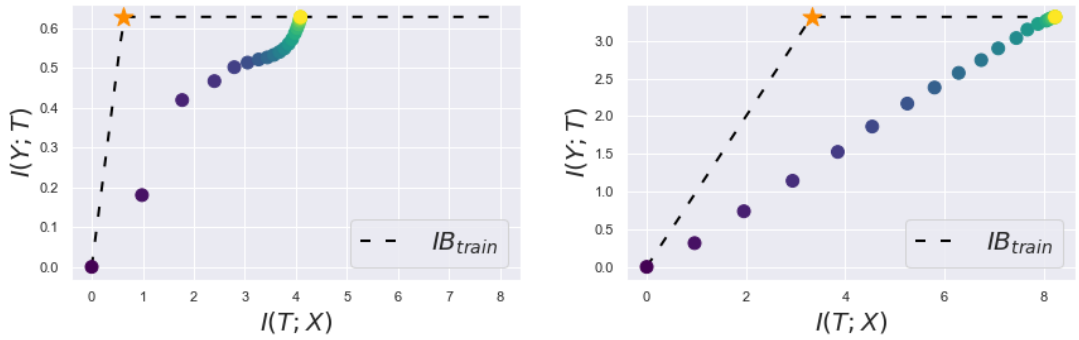
[1] The documentation of the package can be found here : https://github.com/gregversteeg/NPEET

[2] A description of the implementation is provided here: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html

EXPERIMENT I : Perfect fit

For the models which achieve a perfect accuracy on training set, it can be seen on Figure 5.1 that only the last layer representation T_L and the prediction representation $\hat{Y}$ belong to the piecewise linear curve as expected. Moreover, $\hat{Y}$ is located exactly on the IB optimal solution point, and T_L is on the horizontal part part of the curve.

The partition learned for the toy data set is shown on Figure 5.3b. Unsurprisingly, the model imitates perfectly the division between the two regions.



(a) Toy *(b) MNIST*

Figure 5.1: *Given enough complexity, the models are able to achieve a perfect fit to the training data. In such case, the prediction representation of the model must be an IB_{train} optimal solution.*

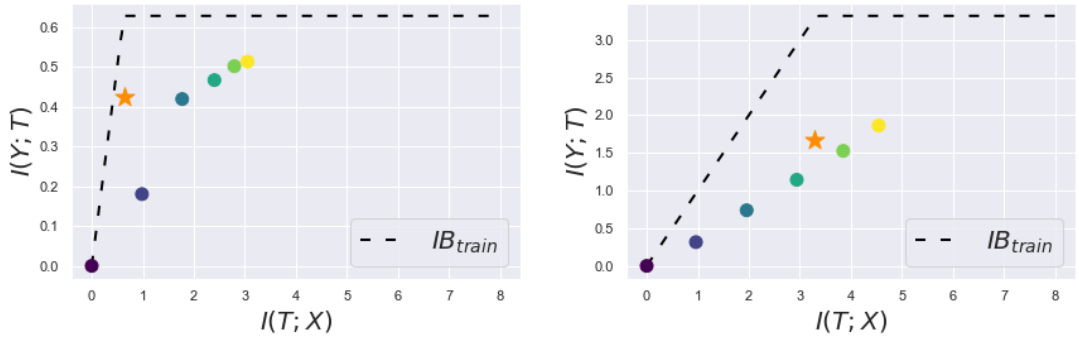
EXPERIMENT II : Imperfect fit

The results are presented on Figure 5.2. Here, the depth of the tree is limited to 5, which is enough to decrease the models accuracy. We observe that no layers attain an IB optimal representation as they are unable to capture enough relevant information given the complexity restriction. In addition, it is observed that relevant information is lost by the prediction transformation f since $\hat{Y}$ is further away from the curve than T_L.

The learned partition for the toy task is shown on Figure 5.3c, from which it can be seen that model does not quiet capture the circle arc.

5.4.2 Evaluating the generalization performances

Three experiments have been carried out. The first one shows how unrepresentative training set impact the the learning dynamics in the information plane. The second



(a) Toy *(b) MNIST*

Figure 5.2: *With limited complexity (depth=5), the models are unable to perfectly classify all samples in the training set. The models obtains a respective correct classification rate of 96,93% and 68,35% on Toy and MNIST. In this case, no layers attain the IB curve.*

(a) Input space. *(b) Perfect fit.* *(c) Imperfect fit.*

Figure 5.3: *Sample space vs. the partitions learned by the model for the toy task with and without depth limit.*

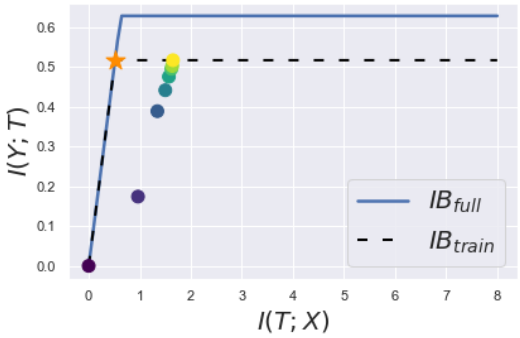
experiment demonstrates that the IB curve contains suboptimal representations, and the last one shows that there exist representations outside the IB curve which generalize better than certain IB optimal representations.

Recall that the generalization performances are measured with respect to the distance of the representation to the IB_{full} optimal solution.

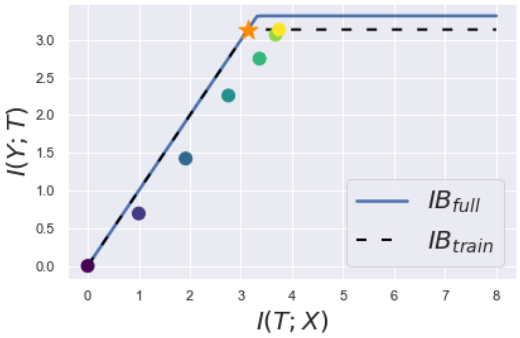
EXPERIMENT I : Non-representative training sets

In order to observe a situation where $IB_{train} \neq IB_{full}$, a very small portion of the full set is used as the training set (1% for toy, and 0.1% for MNIST). Only, it is not possible to observe IB_{train} above IB_{full} for the MNIST dataset because the classes are heavily balanced.

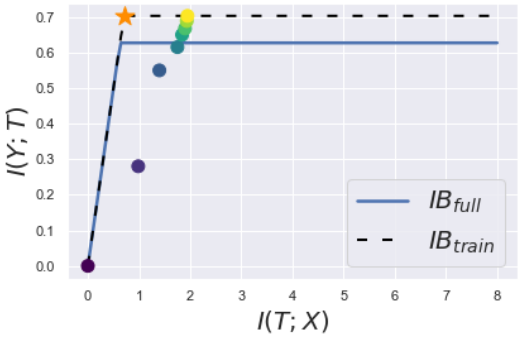
Figure 5.4 shows the results of the experiments. It can be observed that if IB_{train} is under IB_{full}, the model will attain the IB_{train} optimal point for both tasks. The same holds if it is above IB_{full}. All models achieve a perfect accuracy on the training set, but do not on the whole set.



(a) *Toy*



(b) *MNIST*

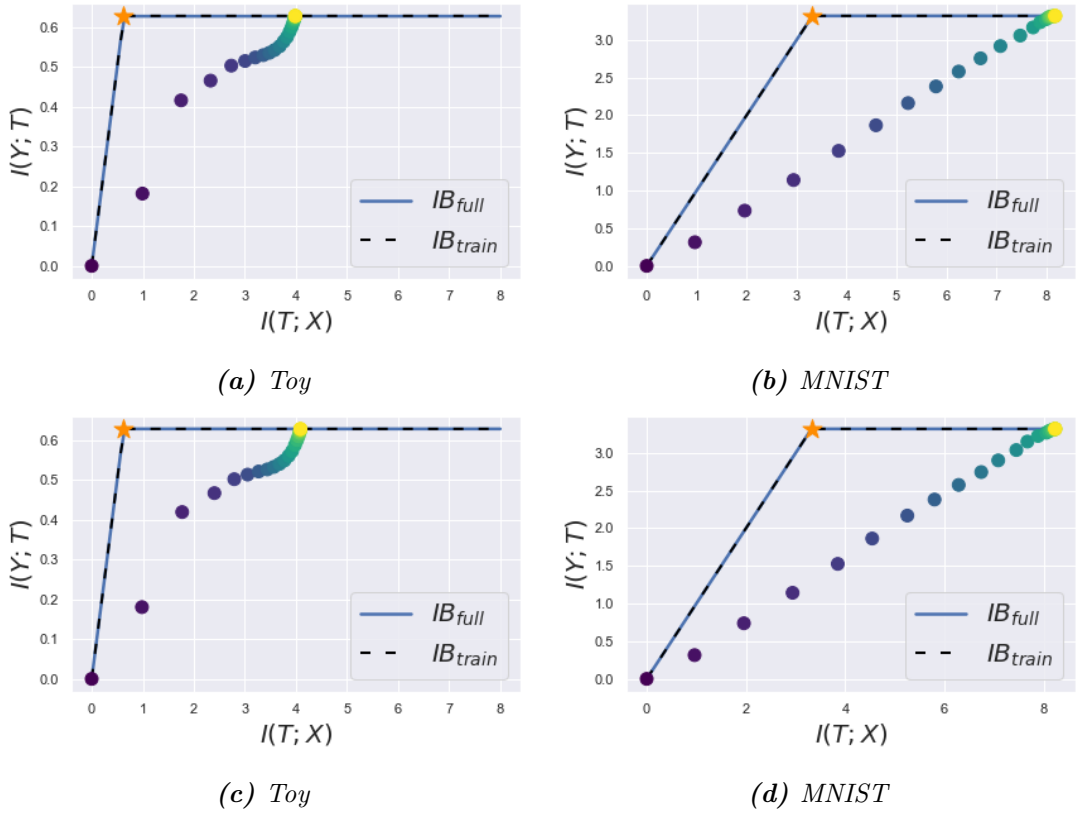


(c) *Toy*

Figure 5.4: *Non-representative training sets. Panel (a) : The model trained on the modified toy dataset reaches a correct classification rate of 95,81% on the full set. Panel (b) : The model trained on the modified MNIST dataset reaches a correct classification rate of 40,1% on the full set. Panel (c) : The model trained on the modified toy dataset reaches a correct classification rate of 94,39% on the full set.*

EXPERIMENT II : The IB curve contains suboptimal representations

A first model is trained on the training set S_{train} and another one on the full set S for both tasks. Figure 5.5 shows that they both are IB_{full} optimal representation even though the former is unable to perfectly classify all samples in the full set S.



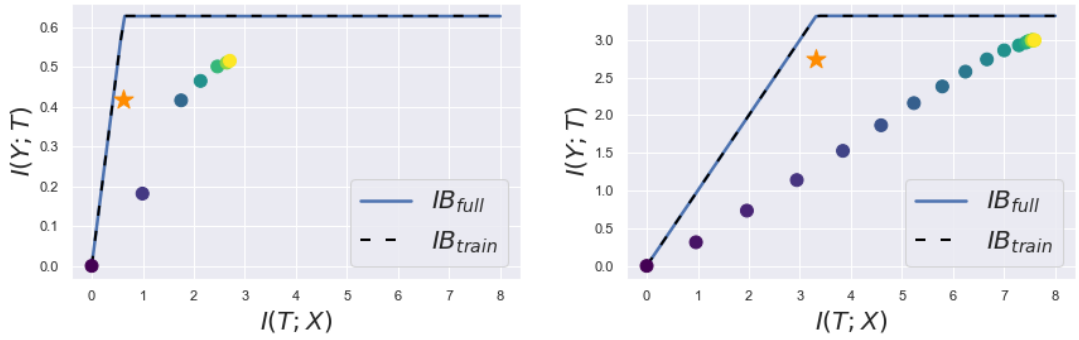
(a) Toy *(b) MNIST*

(c) Toy *(d) MNIST*

Figure 5.5: *The models showcased on panel (a) and (c) are assessed to be generalize equally by the IB, even though the model in (a) achieves a correct classification rate of 99,27% and the one in (c) has one of 100%. The same holds for the models showcased on panel (b) and (d) which attains respectively correct classification rates of 98,36% and 100%.*

EXPERIMENT III : Better representations than IB optimal solution

The models of the previous experiment trained on the training set are pruned using the complexity cost pruning procedure. The α value is selected using a 5-folds cross validation procedure, and correspond to the parameter of pruned model which achieves the lowest estimated expected test error. The results for the selected pruned model are shown on Figure 5.6. It can be seen that the pruned model does not attain the IB_{full} optimal solutions, even though it generalizes better than the non-pruned model.



(a) Toy *(b) MNIST*

Figure 5.6: *Pruned models. Panel (a) : the pruned model attains an correct classification rate of 95,95% on the test set, whereas the non-pruned model obtains a lesser value of 94,89%. Panel (b) the pruned model attains an correct classification rate of 88,57% on the test set, whereas the non-pruned model obtains a lesser value of 88,55%.*

5.5 Summary

In summary :

- The partitions at each depth level of decision trees can be interpreted as intermediate representations in the IB framework. They have the pleasant property of being discrete valued, which removes any risk of unbounded mutual information.
- These successive representations follow an increasing pattern in the information plane, at least until a representation captures all the relevant information. In which case, the tree stops growing and its last layer representation is located on the horizontal part of IB_{train} curve. The prediction map then removes the excessive information, yielding the prediction representation which is located right on the elbow of the curve.
- It has been demonstrated empirically that the IB framework is not adequate to evaluate the generalization performances of decision trees.

Chapter 6

Conclusions

Our goal in this work has been to explore the possibility that the Information Bottleneck could serve as a novel framework in which supervised learning algorithms might be studied, evaluated and compared.

In this regard, the Information Bottleneck method was presented along with all its related tool to study the quality of representations. Then, we reviewed the recent attempts made to understand learning in the IB frameworks. We proposed a novel analysis for the important case of classification task, in which we explored the capabilities of the IB to account for the goodness of a model's fit and its generalization performances. We demonstrated that the IB under delivered in both aspects, as it was showed to be unable to differentiate between models which are perfectly fitted to the training data and models which are not. Moreover, it was also showed that it is unable to distinguish between models with better and worse generalization capabilities.

Next, we introduced the decision tree method in details, as it is the model we chose to conduct empirical investigation of the previous analysis. We then presented to first to its kind IB analysis of decision trees, which was used to perform our experiments, all of which corroborated the previous claims. Moreover, we also showed there exist representation that do are not IB optimal but which generalizes better than optimal IB representations.

Appendix A

IB variations

Alternative formulations of the IB problem have been proposed in order to alleviate alleged deficiencies of the original. Here, we summarize three of those.

Squared-IB

In general settings, the information curve defined in 2.9 can not be explored by varying the tradeoff parameter β of the IB-Lagrangian defined in 2.10 [28]. That is because, the IB-curve is not always strictly concave [27], and so there is no one-to-one correspondence between β and the points belonging to the IB-curve. One such known case arise in deterministic scenario, when Y is a deterministic function of X. Authors of [11] proposed the *squared-IB Lagrangian*

$$\mathcal{L}^{\beta}_{\text{sqIB}}(T) = I(Y;T) - \beta I(X;T)^2 \tag{A.1}$$

and proved that it allows the IB-curve to be explored even when it is merely concave.

Convex-IB

A direct generalization of the squared-IB was proposed in [47]. In it, they consider the family of Lagrangians of the form

$$\mathcal{L}^{\beta_u}_{\text{IB,u}}(T) = I(Y;T) - \beta_u u(I(X;T)) \tag{A.2}$$

with u a monotonically increasing and strictly convex function. Elements of this family are called *convex-IB Lagrangians* which have the property that all points of any IB curve are mapped to a unique β_u.

Clearly, the squared-IB Lagrangian is an element of the convex-IB family with the choice $u(I(X;T)) = I(X;T)^2$.

Other similar families have been proposed in [3].

Deterministic IB (DIB)

Authors of [52] argue that the compression term $I(X;T)$ used in the original IB should be replaced by the quantity $H(T)$. They argue that in the channel coding literature, $I(X;T)$ corresponds to a communication cost whereas $H(T)$ is a representational cost, and so that the latter is a more appropriate penalty. Hence, they propose the following Lagrangian called the *deterministic-IB Lagrangian (DIB)*:

$$\mathcal{L}^{\beta}_{\text{DIB}}(T) = H(T) - \beta I(Y;T) \tag{A.3}$$

The difference between the two Lagrangians turns out to be the conditional entropy

$$\mathcal{L}^{\beta}_{\text{DIB}}(T) - \mathcal{L}^{\beta}_{\text{IB,dual}}(T) = -H(T|X) \tag{A.4}$$

which measures the stochasticity in the mapping from X to T. The minimization of the IB thus corresponds to encouraging stochasticity in the mapping. Hence, as opposed to IB, it was showed that minimizing the DIB Lagrangian favors deterministic encoding distribution.

A generalization of the IB and DIB was also proposed, it is defined by the family of Lagrangian :

$$\mathcal{L}^{\beta}_{\text{IB},\alpha}(T) = H(T) - \alpha H(T|X) - \beta I(Y;T). \tag{A.5}$$

A method to solve optimization problems of this family was also proposed.

In the same paper, they presented experiments which illustrates how IB and DIB can be seen respectively as soft and hard clustering methods. Note that a version of IB which produces hard cluster has been proposed as early as 1999 in [53], called the *Agglomerative information bottleneck*. A key distinction between the two approach is that the Akaike Information Criterion was built to be a hard clustering method whereas the hard clusters in DIB are simply a consequence of the design.

Appendix B

IB solution for a Gaussian joint distribution

In [26], the case of jointly Gaussian distributions was studied and an analytical solution was found. A particularity of the solution is that the dimensionality of T varies continuously with β.

Let (X, Y) be a pair of jointly Gaussian random vectors of dimension n_x, n_y and denote by Σ_x, Σ_y their respective covariance matrix and by $\Sigma_x y$ their cross-covariance matrix. Without loss of generality, assume that X, Y have zero means and Σ_x, Σ_y are full rank. Then, it was proved in [54] that the bottleneck variable T must also be Gaussian and can be represented as the linear projection

$$T = AX + \xi, \qquad \xi \sim \mathcal{N}(0, \Sigma_\xi) \tag{B.1}$$

where $A \in \mathbb{R}^{n_x \times n_x}$ and Σ_ξ is independent of (X, Y).
Moreover, for $\beta \geq 0$, it was proved that the optimum is achieved with the choices $\Sigma_\xi = \mathbb{1}_{n_x}$ (the $n_x \times n_x$ identity matrix) and

$$A(\beta) := \begin{cases} A_0(\beta), & \beta \in [0, \beta_1^c) \\ A_1(\beta), & \beta \in [\beta_1^c, \beta_2^c) \\ \quad\vdots & \\ A_{n_x}(\beta), & \beta \in [\beta_{n_x}^c, +\infty) \end{cases} \tag{B.2}$$

with elements defined as,

$$A_i(\beta) = [\alpha_1(\beta)\mathbf{v}_1, \ldots, \alpha_i(\beta)\mathbf{v}_i, \mathbf{0}, \ldots, \mathbf{0}]^T, \qquad \text{for } i = 1, \ldots, n_x$$

and where

- $(\mathbf{v}_i^T)_{i=1}^{n_x}$ are the left eigenvectors of $\Sigma_{x|y}\Sigma_x^{-1}$ with eigenvalues $(\lambda_i)_{i=1}^{n_x}$ (sorted in ascending order),

- $\beta_i^c = \frac{1}{1-\lambda_i}$ are the critical values of the tradeoff parameter,
- $\alpha_i(\beta) := \sqrt{\frac{\beta(1-\lambda_i)-1}{\lambda_i r_i}}$ with $r_i = \mathbf{v}_i^T \Sigma_X \mathbf{v}_i$,
- $\mathbf{0}$ is the n_x-dimensional zero vector.

As β increase, it goes through a series of critical points, each time increasing the rank of A by an additional eigenvector and so the dimension of the space in which the representation T lives.

Bibliography

[1] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. John Wiley & Sons, Nov. 2012. Google-Books-ID: VWq5GG6ycxMC.

[2] A. Zaidi, I. Estella-Aguerri, and S. Shamai (Shitz), "On the Information Bottleneck Problems: Models, Connections, Applications and Information Theoretic Views," *Entropy*, vol. 22, p. 151, Feb. 2020.

[3] B. Rodriguez Galvez, "The information bottleneck: Connections to other problems, learning and exploration of the ib curve," *Master's Thesis, KTH Royal Institute of Technology, Stockholm, Sweden*, June 2019.

[4] J. Friedman, T. Hastie, and R. Tibshirani, "The elements of statistical learning," *Springer*, 2009.

[5] T. M. Mitchell, "Machine learning," *McGraw Hill*, 1997.

[6] O. Shamir, S. Sabato, and N. Tishby, "Learning and generalization with the information bottleneck," *Theoretical Computer Science*, vol. 411, pp. 2696–2711, June 2010.

[7] A. Alemi, I. Fischer, J. Dillon, and K. Murphy, "Deep Variational Information Bottleneck," in *ICLR*, 2017.

[8] Y. Dubois, D. Kiela, D. J. Schwab, and R. Vedantam, "Learning Optimal Representations with the Decodable Information Bottleneck," Sept. 2020.

[9] N. Zaslavsky, C. Kemp, T. Regier, and N. Tishby, "Efficient compression in color naming and its evolution," *Proceedings of the National Academy of Sciences*, vol. 115, pp. 7937–7942, July 2018.

[10] M. Chalk, O. Marre, and G. Tkacik, "Toward a unified theory of efficient, predictive, and sparse coding," *Proceedings of the National Academy of Sciences*, vol. 115, pp. 186–191, Jan. 2018.

[11] A. Kolchinsky, B. D. Tracey, and D. H. Wolpert, "Nonlinear Information Bottleneck," *Entropy*, vol. 21, p. 1181, Dec. 2019.

[12] M. Chalk, O. Marre, and G. Tkacik, "Relevant sparse codes with variational information bottleneck," *arXiv preprint arXiv:1605.07332*, 2016.

[13] M. Wieser, A. Wieczorek, D. Murezzan, and V. Roth, "Learning Sparse Latent Representations with the Deep Copula Information Bottleneck," 2018.

[14] A. Achille and S. Soatto, "Information Dropout: Learning Optimal Representations Through Noisy Computation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, pp. 2897–2905, Dec. 2018.

[15] A. Wieczorek and V. Roth, "On the Difference between the Information Bottleneck and the Deep Information Bottleneck," *Entropy*, vol. 22, p. 131, Feb. 2020.

[16] A. A. Alemi, I. Fischer, and J. V. Dillon, "Uncertainty in the Variational Information Bottleneck," *arXiv e-prints*, vol. 1807, p. arXiv:1807.00906, July 2018.

[17] 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," Nov. 2016.

[18] N. Slonim and N. Tishby, "Document clustering using word clusters via the information bottleneck method," in *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 208–215, 2000.

[19] N. Slonim, G. S. Atwal, G. Tkacik, and W. Bialek, "Information-based clustering," *Proceedings of the National Academy of Sciences*, vol. 102, pp. 18297–18302, Dec. 2005.

[20] X. L. Li and J. Eisner, "Specializing Word Embeddings (for Parsing) by Information Bottleneck," Sept. 2019.

[21] B. Paranjape, M. Joshi, J. Thickstun, H. Hajishirzi, and L. Zettlemoyer, "An Information Bottleneck Approach for Controlling Conciseness in Rationale Extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Online), pp. 1938–1952, Association for Computational Linguistics, Nov. 2020.

[22] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal 27.3*, 1948.

[23] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," Apr. 2000.

[24] R. Gilad-Bachrach, A. Navot, and N. Tishby, "An Information Theoretic Tradeoff between Complexity and Accuracy," in *Learning Theory and Kernel Machines* (B. Schölkopf and M. K. Warmuth, eds.), Lecture Notes in Computer Science, (Berlin, Heidelberg), pp. 595–609, Springer, 2003.

[25] H. Hafez-Kolahi and S. Kasaei, "Information Bottleneck and its Applications in Deep Learning," Apr. 2019.

[26] G. Chechik, A. Globerson, N. Tishby, Y. Weiss, and P. Dayan, "Information Bottleneck for Gaussian Variables.," *Journal of machine learning research*, vol. 6, no. 1, 2005.

[27] H. Witsenhausen and A. Wyner, "A conditional entropy bound for a pair of discrete random variables," *IEEE Transactions on Information Theory*, vol. 21, pp. 493–501, Sept. 1975.

[28] A. Kolchinsky, B. Tracey, and S. Van Kuyk, "Caveats for information bottleneck in deterministic scenarios," Feb. 2019.

[29] R. Blahut, "Computation of channel capacity and rate-distortion functions," *IEEE transactions on Information Theory*, vol. 18, no. 4, pp. 460–473, 1972.

[30] N. Tishby and N. Zaslavsky, "Deep learning and the information bottleneck principle," in *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, Apr. 2015.

[31] R. Shwartz-Ziv and N. Tishby, "Opening the Black Box of Deep Neural Networks via Information," Apr. 2017.

[32] A. M. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. D. Tracey, and D. D. Cox, "On the information bottleneck theory of deep learning," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, p. 124020, Dec. 2019.

[33] R. A. Amjad and B. C. Geiger, "Learning Representations for Neural Network-Based Classification Using the Information Bottleneck Principle," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 2225–2239, Sept. 2020.

[34] Z. Goldfeld and Y. Polyanskiy, "The Information Bottleneck Problem and its Applications in Machine Learning," *IEEE Journal on Selected Areas in Information Theory*, vol. 1, pp. 19–38, May 2020.

[35] P. Khadivi, R. Tandon, and N. Ramakrishnan, "Flow of Information in Feed-Forward Deep Neural Networks," Mar. 2016.

[36] K. Wickstrøm, S. Løkse, M. Kampffmeyer, S. Yu, J. Principe, and R. Jenssen, "Information Plane Analysis of Deep Neural Networks via Matrix-Based Renyi's Entropy and Tensor Kernels," *arXiv:1909.11396 [cs, stat]*, Sept. 2019. arXiv: 1909.11396.

[37] A. Achille and S. Soatto, "Emergence of Invariance and Disentanglement in Deep Representations," in *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9, Feb. 2018.

[38] M. Gabrié, A. Manoel, C. Luneau, J. Barbier, N. Macris, F. Krzakala, and L. Zdeborová, "Entropy and mutual information in models of deep neural networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2019, p. 124014, Dec. 2019.

[39] B. C. Geiger, "On Information Plane Analyses of Neural Network Classifiers – A Review," June 2021.

[40] H. Cheng, D. Lian, S. Gao, and Y. Geng, "Utilizing Information Bottleneck to Evaluate the Capability of Deep Neural Networks for Image Classification," *Entropy*, vol. 21, p. 456, May 2019.

[41] Y. Song, L. Yu, Z. Cao, Z. Zhou, J. Shen, S. Shao, W. Zhang, and Y. Yu, "Improving Unsupervised Domain Adaptation with Variational Information Bottleneck," *arXiv preprint arXiv:1911.09310*, 2019.

[42] T. T. Nguyen and J. Choi, "Layer-wise Learning of Stochastic Neural Networks with Information Bottleneck," *Entropy*, vol. 21, p. 976, Oct. 2019.

[43] B. Dai, C. Zhu, B. Guo, and D. Wipf, "Compressing Neural Networks using the Variational Information Bottleneck," in *International Conference on Machine Learning*, pp. 1135–1144, PMLR, July 2018.

[44] D. Russo and J. Zou, "How much does your data exploration overfit? Controlling bias via information usage," Oct. 2019.

[45] M. Vera, P. Piantanida, and L. R. Vega, "The Role of the Information Bottleneck in Representation Learning," in *2018 IEEE International Symposium on Information Theory (ISIT)*, pp. 1580–1584, June 2018. ISSN: 2157-8117.

[46] A. Xu and M. Raginsky, "Information-theoretic analysis of generalization capability of learning algorithms," *arXiv preprint arXiv:1705.07809*, 2017.

[47] B. Rodríguez Gálvez, R. Thobaben, and M. Skoglund, "The Convex Information Bottleneck Lagrangian," *Entropy*, vol. 22, p. 98, Jan. 2020.

[48] M. Noshad, Y. Zeng, and A. O. Hero, "Scalable Mutual Information Estimation Using Dependence Graphs," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2962–2966, May 2019. ISSN: 2379-190X.

[49] Z. Goldfeld, E. V. D. Berg, K. Greenewald, I. Melnyk, N. Nguyen, B. Kingsbury, and Y. Polyanskiy, "Estimating Information Flow in Deep Neural Networks," in *International Conference on Machine Learning*, pp. 2299–2308, PMLR, May 2019.

[50] B. C. Geiger, "On Information Plane Analyses of Neural Network Classifiers–A Review," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[51] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees," *Routledge*, 2017.

[52] D. Strouse and D. J. Schwab, "The Deterministic Information Bottleneck," *Neural Computation*, vol. 29, pp. 1611–1630, June 2017.

[53] N. Slonim and N. Tishby, "Agglomerative information bottleneck," *NIPS. Vol. 4*, 1999.

[54] A. Globerson and N. Tishby, "On the optimality of the gaussian information bottleneck curve," *The Hebrew University of Jerusalem, Tech. Rep*, 2004.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**