



The Mathematics of the Unknown

Bringing to the present the knowledge of the future

R. A. García Leiva

(This book is 80% complete)

Copyright © 2023 R. A. García Leiva
www.mathematicsunknonw.com

PUBLISHED BY THE AUTHOR

To my wife Justi, my son Daniel,
and my two daughters Teresa and Lucía.



Contents

1	Introduction	21
1.1	Entities	22
1.2	Representations	23
1.3	Descriptions	25
1.4	Miscoding	26
1.5	Inaccuracy	27
1.6	Surfeit	29
1.7	Nescience	31
1.8	Evolution of Nescience	33
1.9	Other Metrics	35
1.10	Interesting Research Questions	37
1.11	New Research Entities	39
1.12	References and Further Reading	40

I

Part 1: Background

2	Discrete Mathematics	43
2.1	Sets, Relations and Functions	44
2.2	Strings and Languages	47
2.3	Counting Methods	49
2.4	Matrices	50
2.5	Graphs	52

3	Discrete Probability	55
3.1	Foundations	56
3.2	Conditional Probability	59
3.3	Random Variables	62
3.3.1	Multivariate Distributions	65
3.3.2	Marginal Distribution	66
3.3.3	Conditional Distributions	67
3.4	Random Samples	68
3.5	Characterizing Distributions	68
3.5.1	Measures of Central Tendency	68
3.5.2	Measures of Dispersion	70
3.5.3	Measures of Statistical Relationship	71
3.6	Common Distributions	72
3.6.1	Uniform Distribution	72
3.6.2	Bernoulli Distributions	72
3.6.3	Binomial Distributions	72
3.7	Large Random Samples	73
3.7.1	Law of Large Numbers	73
3.7.2	Central Limit Theorem	75
4	Computability	77
4.1	Turing Machines	78
4.2	Universal Turing Machines	81
4.3	Non-Computable Problems	82
4.4	Computable Functions and Sets	84
4.5	Oracle Turing Machine	85
4.6	Computational Complexity	86
5	Coding	91
5.1	Coding	92
5.2	Kraft Inequality	95
5.3	Optimal Codes	98
5.4	Entropy	101
5.5	Huffman Algorithm	104
5.6	Discretization Algorithms	106
5.6.1	k-means Clustering	107
6	Complexity	109
6.1	Strings Complexity	110
6.2	Properties of Complexity	112
6.3	Joint Kolmogorov Complexity	113
6.4	Conditional Kolmogorov complexity	114
6.5	Information Distance	116
6.6	Incompressibility and Randomness	118

7	Learning	121
7.1	Statistical Inference	121
7.1.1	Bayesian Inference	123
7.1.2	Non-Bayesian Inference	124
7.2	Machine Learning	125
7.2.1	Model Accuracy	126
7.2.2	No free lunch theorem	127
7.2.3	The bias-variance trade-off	127
7.2.4	Generative vs. discriminative models	127
7.2.5	Decision Trees	127
7.2.6	Time Series Analysis	128
7.3	Minimum Message Length	132
7.4	Minimum Description Length	134
7.4.1	Refined MDL	135
7.5	Multiobjective Optimization	136
7.5.1	Range of the Solutions	137
7.5.2	Trade-offs	138
7.5.3	Optimization Methods	139
8	Philosophy of Science	143
8.1	Metaphysics	143
8.2	Scientific Representation	144
8.3	Models in Science	145
8.4	Scientific Theories	146
8.5	The Scientific Method	146
8.6	Scientific Discovery	147

II

Part 2: Foundations

9	Entities, Representations and Descriptions	151
9.1	Entities	152
9.2	Representations	153
9.3	Invalid Representations	157
9.4	Joint Representations	158
9.5	Descriptions	160
9.6	Descriptions for Joint Representations	163
9.7	Conditional Descriptions	164
9.8	Research Areas	166
9.9	References	168
10	Miscoding	169
10.1	Miscoding	170
10.2	Joint Miscoding	171

10.3	Decreasing Miscoding	173
10.4	Targetless Representations	173
10.5	Miscoding of Areas	175
11	Inaccuracy	177
11.1	Inaccuracy	178
11.2	Conditional Inaccuracy	179
11.3	Decreasing Inaccuracy	181
11.4	Inaccuracy-Miscoding Rate of Change	183
11.5	Inaccuracy of Areas	184
12	Surfeit	187
12.1	Surfeit	188
12.2	Conditional Surfeit	190
12.3	Decreasing Surfeit	191
12.4	Rate of Change	191
12.5	Surfeit of Areas	191
13	Mismodel	193
13.1	Mismodel	193
13.1.1	Mismodel of Joint Representations	194
13.1.2	Conditional Mismodel	194
13.2	Reducing Mismodel	195
13.3	Mismodel of Areas	196
14	Nescience	197
14.1	Nescience	198
14.2	Minimizing Nescience	202
14.2.1	Global Criterion	202
14.2.2	Weighting Method	203
14.2.3	The other method	203
14.2.4	Evolutionary Methods	203
14.3	Joint Nescience	203
14.4	Conditional Nescience	204
14.5	Nescience of Areas	204
14.6	Perfect Knowledge	205
14.7	Current Best Description	207
14.8	Nescience based on Datasets	207
14.9	Unknonwn Unknown	208
15	Interesting Questions	209
15.1	Relevance	210
15.2	Applicability	212

15.3	Interesting Questions	214
15.4	New Research Topics	215
15.5	Classification of Research Areas	217
16	Advanced Properties	219
16.1	The Axioms of Science	219
16.1.1	Model Theory	220
16.1.2	Type Theory	223
16.1.3	Category Theory	230
16.2	Scientific Method	230
16.2.1	Science as a Language	230
16.3	The Inaccuracy - Surfeit Trade-off	231
16.4	Science vs. Pseudoscience	231
16.5	Graspness	232
16.6	Effort	232
16.7	Human Understanding	232
16.8	Areas in Decay	233

III

Part 3: Applications

17	Machine Learning	237
17.1	Nescience Python Library	237
17.2	A Note About Compression	238
17.3	Miscoding	240
17.4	Inaccuracy	247
17.5	Surfeit	250
17.6	Nescience	253
17.7	Auto Machine Classification	257
17.7.1	Surfeit of Algorithms	258
17.8	Auto Machine Regression	258
17.9	Time Series	258
17.9.1	Automiscoding, Crossmiscoding and Partial Automiscoding	258
17.9.2	Auto Time Series	261
17.10	Anomaly Detection	262
17.11	Decision Trees	264
17.11.1	Algorithm Description	265
17.11.2	Algorithm Evaluation	268
17.12	Algebraic Model Selection	272
17.13	The Analysis of the Incompressible	275
18	Software Engineering	277
18.1	Redundancy of Software	277

18.2	Quality Assurance	281
18.3	Forex Trading Robots	281
19	Philosophy of Science	285
19.1	Wikipedia, The Free Encyclopedia	285
19.2	Classification of Research Topics	285
19.3	Classification of Research Areas	290
19.4	Interesting Research Questions	290
19.5	Interesting Research Topics	292
19.6	Philosophy of Science	294
19.7	Evolution of Knowledge	297
19.8	Graspness of Topics	298
19.9	Probability of Being True	298
19.10	Unused text	299
20	Computational Creativity	301
20.1	Classification of Research Topics	302
20.1.1	Relevance	302
20.1.2	Applicability	303
20.1.3	Maturity	304
20.2	Interesting Research Questions	304
20.2.1	Intradisciplinary Questions	304
20.2.2	Interdisciplinary Questions	304
20.3	New Research Topics	304
20.4	Classification of Research Areas	304
20.5	References	304
20.6	Future Work	305

IV

Appendix

A	Linear Algebra	309
A.1	Vectors	309
A.2	Matrices	309
B	Foundations of Mathematics	311
B.1	Propositional Logic	311
B.2	Predicate Logic	314
B.3	Set Theory	315
B.4	Lambda Calculus	316
B.5	Category Theory	317

C	Advanced Mathematics	319
C.1	Distinctiveness of Metrics	319
C.2	Distinctiveness of Metrics	320
C.2.1	Accuracy	320
C.2.2	Precision	320
C.2.3	Recall	321
C.2.4	F1	321
C.2.5	Area under the ROC curve	322
D	Coq Proof Assistant	325
E	About Quotes and Photos	329
E.0.1	Quotes	329
E.0.2	Photos	331
F	Notation	335
	Bibliography	339
	Books	339
	Articles	339



Preface

*Perfection is achieved not when there is nothing more to add,
but when there is nothing left to take away.*

Antoine de Saint-Exupéry

The core premise of this book asserts that perfect knowledge implies randomness. This notion may seem highly counterintuitive at first glance, given that much of scientific endeavor focuses on naming, organizing, and classifying our intricate and chaotic world. It would appear that science is anything but random. However, that is not the case.

A lengthy explanation of a scientific concept often indicates an incomplete understanding. The calculus of derivatives serves as a fitting example. In the era of Newton and Leibniz, comprehending the concept of a function's derivative necessitated substantial space for definition and was grasped only by a select group of specialists. In contrast, today's definition of derivative is concisely explained in a single paragraph and taught in high schools.

Long explanations tend to be superfluous, filled with redundancies, extraneous concepts, improperly identified relationships, and poor notation. As our understanding deepens, often through extensive research, we can eliminate the unnecessary elements from theories, eventually achieving a state of perfect knowledge. When a theory is perfect and nothing remains to be removed, its description becomes an incompressible string, which aligns with the mathematical definition of a random string - an incompressible sequence of symbols. A scientific theory is deemed random when it encompasses the maximum amount of information within the smallest possible space.

Randomness sets the ultimate limit to our knowledge, since a random description cannot be refined any further and, assuming the theory is accurate, our understanding must be perfect. Far from being a handicap, the constraints imposed by randomness to knowledge pave the way for new possibilities in science and technology. By comprehending these limitations, we can address the most challenging open problems and discover new research questions. Indeed, this understanding can lead us to innovative approaches and a deeper appreciation for the inherent complexities of scientific discovery.

This book introduces the new "Theory of Nescience," a mathematical framework that reexamines the nature of science and the acquisition of scientific knowledge. We begin with the assumption that it is easier to measure our ignorance than to quantify our understanding, as randomness sets the ultimate boundary on our knowledge. We also posit that the computer, as a conceptual model, is the ideal tool to measure this elusive quantity. The book also explores the practical applications of the theory of nescience in scientific research, artificial intelligence, computational creativity, and software engineering.

Research Agenda

In this section we present a comprehensive list of research questions that aim to address critical gaps in our current understanding of science and the scientific method. With the objective of fostering intellectual curiosity and scientific progress, our interdisciplinary approach will bring together knowledge from multiple and diverse fields, combining their unique perspectives and methods. In pursuit of answers to these pressing questions, we aim to expand the boundaries of human understanding and contribute to the development of innovative solutions that have the potential to transform our society and the world around us.

Q1: Can we provide a quantitative characterization of our ignorance regarding a research topic? Utilizing this metric would enable us to not only gauge the extent of our lack of knowledge on a specific subject, such as climate change, but also to measure the degree to which a new development or idea (typically published in a research paper) contributes to enhancing our understanding. By combining this metric with an assessment of the problem's relevance, we could effectively quantify the value of new scientific contributions.

Q2: Can we compare the extent of our ignorance across disparate scientific fields? If feasible, this would enable us to classify and compare all open questions based on the extent of our lack of knowledge, independent of the disciplines they belong to (e.g., physics, biology, sociology). Coupling this classification with the previously mentioned relevance metric would allow us to determine where to focus our research efforts. It is important to note that this proposal does not suggest ceasing research in basic or fundamental science; on the contrary, it underscores the necessity of such inquiries.

Q3: How can we differentiate between science and pseudo-science? This enduring and unresolved question in the philosophy of science has significant implications. Developing a practical, mathematically-based solution would be highly beneficial, as it would enable us to evaluate the scientific nature of various controversial disciplines that claim to be scientific.

Q4: Do some research topics inherently possess a higher degree of complexity than others? Addressing this question would help determine whether researchers in certain disciplines are genuinely more intellectually adept than their counterparts in other fields, as they often claim, or if the perceived difference is merely a consequence of some subjects being easier to comprehend than others.

Q5: Are there research topics beyond the scope of human comprehension? What are the boundaries of human understanding? It is possible that certain problems may be unsolvable by humans, and some research topics could be beyond the grasp of our limited cognitive abilities. It might be necessary to accept that progress in specific research areas can only be achieved by relinquishing control and allowing computers to perform the creative scientific work.

Q6: Can we establish a systematic procedure to enhance our knowledge? It is possible that the so-called "scientific method" represents an unsolvable problem, with only practical approximations available to us. If this is the case, how can we evaluate and compare different approximations? Could the principles of the theory of nescience be employed to develop a novel and more effective method for expanding our understanding?

Q7: What effort is required to fully comprehend an unfamiliar subject? For instance, what

would be the cost of increasing our understanding of cancer treatment by 1%? With such a metric, we could strategize our research activities based on priorities, budget, and the significance of potential outcomes. Naturally, this metric would represent a lower bound, as the actual effort could be greater due to factors such as poor project management.

Q8: What constitutes perfect knowledge? Is perfect knowledge attainable for all possible entities? Randomness serves as a necessary condition for perfect knowledge, but additional criteria are required to fully define complete understanding. Furthermore, it is essential to determine whether perfect knowledge is universally achievable or if there are inherent limitations that render perfection in science an unattainable ideal.

Q9: Can we devise a method for discovering new, previously unknown, and intriguing research entities and problems? A procedure is needed to explore the unknown unknowns—problems that we not only lack solutions for but are also unaware of their existence. Such a method could enable us to uncover future research topics and bring them to the forefront of current investigation.

Some responses presented in this book are more developed than others. For certain questions, we will provide a comprehensive theoretical answer accompanied by a practical implementation, while for others, we will offer only a rough outline of a potential solution. However, we are confident that with further research, the new theory of nescience can yield satisfactory answers to all the questions posed. In this regard, this book serves more as a research agenda rather than a complete description of a fully-developed theory of nescience.

Origins of the Theory of Nescience

It was in 1991, when I was eighteen years old, that I first encountered the statement, "*Computers are useless, they can only give you answers*". This quote is attributed to Pablo Picasso, one of the most creative and influential artists of the 20th century. I quickly realized the profound truth in his words, as it is indeed accurate that computers cannot generate original and interesting questions. However, it wasn't until more than 20 years later, in 2014, that I began to seriously consider the challenge Picasso's observation presented to the computer science community.

Much of the foundation for my methodology in discovering interesting questions emerged during a single sleepless night: nescience, relevance, the unknown unknown, and various other novel concepts. It's possible that my subconscious mind had been developing these ideas over the course of 20 years, as I believe it's no coincidence that I chose to study subjects like information theory and Kolmogorov complexity long before I realized their significance to my future theory of nescience. Remarkably, it took just one night to conceive a rough outline of the key ideas, a couple of months to conduct initial computer experiments that validated them, and several years to develop the necessary mathematics for a robust theoretical framework.

Initially, my primary interest was in uncovering interesting scientific questions, delving into what I call the unknown unknown area to reveal future research topics. As my focus shifted to the evolution of nescience in scientific subjects over time, I began to ponder the implications of perpetually refining a theory. That's when I discovered the connection between perfect knowledge and randomness, which led to the expansion of my original methodology for identifying interesting questions into a comprehensive theory of nescience that further explored this crucial concept.

I must admit that the new theory received a warm reception from my colleagues and other scientists with whom I had the chance to discuss it during the initial development stages. This early success spurred me to refine the core ideas and their practical applications. While developing the mathematics underpinning the concept of nescience, I began exploring other potential applications, such as analyzing computer programs for quality assurance and examining raw datasets for machine learning purposes.

Despite the explanatory power of the theory, I remained unsatisfied with its status. It's true that the theory enabled me to address some open questions in the philosophy of science, such as the

extent to which we understand mathematics compared to sociology, why certain research topics are more challenging than others, and the fundamental differences between science and pseudoscience, among others. However, the problem was that the theory did not yield any predictions that could be falsified through experimentation. As a result, I chose to take a risk and further develop the mathematics to generate such predictions. This led to the creation of a function that describes how nescience decreases as research effort increases. This function allows us to predict the maximum increase in knowledge about a topic based on the amount of effort we are willing to invest in understanding it.

Yet, I still wasn't entirely satisfied, so I decided to push the theory even further and seek a novel prediction—something that had not been observed before. New advancements in the mathematical foundations of the theory enabled me to uncover a highly counterintuitive property: sometimes, for certain classes of topics, additional research can be counterproductive. In other words, the more research we do, the more confused we become, as it is not possible to increase our knowledge beyond a critical point for these topics, even if that point is far from perfect knowledge.

About the Book

The theory of nescience draws upon concepts from various academic disciplines, including computability, randomness, information theory, complexity, probability, graph theory, philosophy of science, and many more. Nevertheless, this book is self-contained, requiring only a basic understanding of first-year calculus, linear algebra, and some programming experience. The content is designed to cater to readers with diverse backgrounds, such as mathematicians, computer scientists, engineers, etc. The mathematical level is suitable for graduate students or advanced undergraduates.

The book is structured into three main parts: Background, Foundations, and Applications. Readers already familiar with the mathematics covered in the Background section may proceed directly to the Foundations part. However, it is highly recommended to at least briefly review the notation used in these chapters. Once acquainted with the details of the theory of nescience described in the Foundations section, the reader can move on to the Applications. A comprehensive understanding of the theory is not necessary to grasp the applications; a basic knowledge of the main concepts and results should suffice.

- *Chapter 1 Introduction* offers a gentle introduction to the theory of nescience, along with a brief overview of the main results. The chapter avoids the use of advanced mathematics, but the concepts are still introduced semi-formally. Although not advised, readers who find it difficult to follow the mathematics behind the theory can read only this first chapter, the introductory sections of the remaining chapters in the Background and Foundations parts, and then proceed directly to the Applications.

PART I Background serves as an introduction to the mathematics needed to measure our lack of knowledge about a research topic and the randomness of a string. The goal of this part is to establish notation, formally define concepts, and prove significant results. Although no prior knowledge is assumed for following the material, it is recommended to consult the standard literature (refer to the References section at the end of each chapter) for a thorough understanding of these topics. Additional subjects are covered in the appendices.

- *Chapter 2 Discrete Mathematics* offers a summary of the fundamental of discrete mathematics needed to grasp the more advanced topics discussed in the book. This chapter serves as a quick review of these concepts, without providing formal definitions or proving theorems. Topics covered include sets, relations, strings, graphs, vectors, matrices and counting methods.
- *Chapter 3 Discrete Probability* provides an introduction to the foundational concepts of probability of discrete events. Topics covered include conditional probability, random variables, characterizing distributions, common distributions, and large random samples.

This chapter aims to equip readers with the necessary background in probability to understand more advanced mathematics presented in the book.

- *Chapter 4 Computability* presents a formal definition of the concept of algorithm. We introduce the idea of a universal Turing machine and demonstrate that certain well-defined mathematical problems cannot be solved by computers. The essential tools of oracle Turing machines and Turing reducibility are also examined in detail. Key results in computational complexity are briefly reviewed.
- In *Chapter 5 Coding*, we explore the properties of codes and, specifically, how codes enable us to compress text without losing information by removing statistical redundancy. We will see that there is a limit to how much text can be compressed using this technique, and that this limit is determined by the entropy of the source. The relation between optimal codes and discrete probabilities is also covered.
- *Chapter 6 Complexity* introduces an absolute metric, called Kolmogorov complexity, to measure the amount of information contained in a string by calculating the length of the shortest computer program that can print that string. The properties of this metric are studied in detail. The relationship between string complexity and randomness is also discussed.
- *Chapter 7 Learning* investigates the relationship between codes and probabilities. It also provides a concise overview of the field of statistical learning, focusing on existing approaches that apply the concept of minimum string length to the problem of stochastic model evaluation and optimal parameter selection. An introduction to the concepts and notation of nonlinear multiobjective optimization problems is also included.
- *Chapter 8 Philosophy of Science* is a brief introduction to the field of philosophy of science. We will review concepts such as scientific representations, models, theories, and other aspects of science from a philosophical perspective, identifying the crucial elements that any formal theory of science should encompass. The chapter also contains an overview of the current state of the art regarding the scientific method.

PART II Foundations offers a comprehensive presentation of the theory of nescience, including formal definitions of its concepts and proofs of the key theoretical results. This part of the book represents the core of the new theory. Readers with a strong background in computability, complexity, information theory, and probability may proceed directly to this part.

- *Chapter 9 Entities, Representations, and Descriptions* introduces the fundamental components of the theory of nescience: entities, representations, and descriptions. The properties of these elements and their interrelationships are examined. The chapter explores how multiple representations and descriptions can be combined, the incorporation of background knowledge in research, the connection between perfect knowledge and randomness, and proposes a novel concept of research area.
- *Chapter 10 Miscoding* addresses the challenging task of representing abstract and non-abstract research entities as strings of symbols for research purposes. The chapter formally introduces the concept of miscoding and investigates its properties. Miscoding quantifies the error introduced due to improper encodings.
- *Chapter 13 Mismodel* introduces a new metric called mismodel. The purpose of the concept of mismodel is to provide a unified framework for understanding and evaluating the quality of descriptions or models in the context of research. By combining inaccuracy and surfeit, mismodel offers a comprehensive assessment of how well a description or model captures the essential features of an entity while avoiding unnecessary complexity or redundancy.
- *Chapter 14 Nescience* is the central chapter of the book, containing the mathematical foundations of the new theory. The chapter formally defines the concept of nescience and studies its main properties, such as the evolution of nescience over time, the meaning of perfect knowledge, and the identification of our current best model.

- *Chapter 15 Interesting Questions* outlines a procedure for identifying interesting aspects within a large collection of measurable objects, particularly for discovering new research questions and topics. New concepts such as relevance and applicability are defined and investigated.
- *Chapter 16 Advanced Properties* delves into advanced concepts and properties of the theory of nescience. While not essential for understanding the theory's applications, readers are encouraged to explore these properties during a subsequent reading of the book. Among the new concepts introduced are an axiomatic version of the theory, graspness as a measure of a research topic's difficulty, and the minimum effort required to reduce a topic's nescience.

PART III Applications presents a collection of practical applications of the concept of nescience in areas such as machine learning, software engineering, philosophy of science, and computational creativity. These applications have been chosen to encompass the entire spectrum of potential topics, ranging from abstract ones (research topics) to real objects and strings (computer programs).

- *Chapter 17 Machine Learning* explores how the concept of nescience can be applied to entities represented by large datasets. Specifically, the methodology will be used to select relevant features, identify optimal models, and compute errors. Additionally, new machine learning algorithms, such as a novel approach to deriving optimal decision trees, will be proposed.
- *Chapter 18 Software Engineering* discusses the use of the theory of nescience with computer programs in order to measure our understanding of current software platforms (operating systems, networking middleware, productivity tools, etc.). We will evaluate whether current software versions are superior to past versions or if software quality is deteriorating over time. The results will also be applied to the automatic discovery of errors, with the aim of improving software quality.
- *Chapter 19 Philosophy of Science* examines the application of the new metrics introduced in this book to the study of science and its methods. Some significant open problems in the field of philosophy of science will be addressed, including the demarcation problem (distinguishing science from pseudoscience) and the question of how science progresses. Various proposals of the scientific method will be evaluated in the context of the theory of nescience and compared to our own proposal.
- *Chapter 20 Computational Creativity* demonstrates the practical application of the methodology for discovering interesting things. In particular, the methodology will be applied to find new research questions and topics, with multiple examples of interesting questions and research topics provided.

Appendix B Foundations of Mathematics describes three distinct theoretical frameworks for formalizing mathematics: logic and set theory, dependent type theory, and category theory. These frameworks can also serve as a solid foundation for the theory of nescience.

Appendix C Advanced Mathematics provides a concise overview of some mathematical concepts used in the book that play a secondary role in the development of the theory. These concepts are included in this appendix for reference.

Appendix D Mechanical Proofs offers a mechanical implementation of the axioms of the theory of nescience and its most significant results, based on the calculus of inductive constructions and, specifically, the Coq proof assistant.

Lastly, *Appendix E About the Photos* explains the origin and intended meaning of the carefully selected photographs featured at the beginning of each chapter.

Acknowledgements

I would like to express my gratitude to everyone who has contributed their comments and ideas to the development of the theory of nescience. In particular, I am grateful to Antonio Fernández, Vincenzo Mancuso, and Paolo Casari, who believed in and supported this project from its very beginning when it was merely a far-fetched idea (although it may still be). Others who have provided contributions and valuable feedback include Héctor Cordobés, Luis F. Chiroque, Agustín Santos, Marco Ajmone, Pablo Rojo, Manuel Cebrián, Andrés Ortega, Emilio Amaya, Mattis Choumanivong, Alexander Lynch, Andrés Carrillo, and Simon Bihoreau. The `fastautoml` library described in Chapter 17 has been partially funded by the IMDEA Networks Institute, the European Union's Horizon 2020 research and innovation programme under grant agreement No 732667 RECAP, and Nokia Spain through the project NetPredict. Also, I would like to extend my thanks to OpenAI/ChatGPT for their valuable contributions in writing this book, as their assistance has played an important role in writing the text describing the ideas presented in it.

Lastly, I am grateful to my parents, who provided me the opportunities they never had, and to my wife and three children, who give my life fundamental meaning.

Disclaimer

I want to clarify that the vast majority of ideas presented in this book are not my own. The inspiration comes from the brilliant minds of our history, such as Occam, Llull, Leibniz, Newton, and philosophers like Plato, Popper, Feyerabend, Wittgenstein, among others. Additionally, I have built upon the mathematical theories developed by scientists like Turing, Church, Post, Shannon, Solomonoff, Chaitin, Kolmogorov, and many more. My sole original contribution with this book may be connecting some dots and offering a potentially interesting reinterpretation of some existing ideas. The References sections at the end of each chapter contain descriptions of the works I have relied upon while writing the book. In the text, I use passive voice ("it is defined") when I know that a concept is not mine, and active voice ("we define") when I am not aware of a previous use of the concept by someone else.



1. Introduction

*If presented with a choice between indifferent alternatives,
then one ought to select the simplest one.*

Occam's razor principle

The pursuit of knowledge typically begins with the identification of a collection of entities we seek to comprehend. The constituents of this collection can be extraordinarily diverse. For instance, mathematicians may focus on mathematical concepts, biologists on living organisms, and engineers on problem-solving machines. Our objective, as researchers, is to gain as much understanding of these entities as possible. This understanding, which allows us to predict the outcomes of our actions, is crucial. For example, we know that applying sufficient heat to a pile of wood will ignite a fire. More challengingly, by observing effects, we can attempt to deduce their causes; a fever may indicate a viral infection. Practical problem-solving hinges on understanding, which entails discerning patterns or regularities enabling us to construct a simplified description or model of the original entities. Such models facilitate our manipulation of these entities for our benefit.

Ideally, we aspire to construct descriptions that allow us to reconstruct the original entities under study. However, for a vast number of entities, particularly abstract ones, this is not achievable. Consequently, we must resort to working with representations - using texts or data that strive to capture as many details of the original entities as feasible. For instance, if we are physicists, an entity's representation could be the outcome of an experiment; for computer scientists, it might be a dataset composed of measurements, and for sociologists, it could be a collection of observed facts. Figure 1.1 illustrates this process: while we desire our descriptions to model entities, in reality, they model our artificially constructed representations of those entities.

The fundamental question lies in how effectively our descriptions interpret the original entities through the modelling of the representations that encode those entities. We must take into account various potential sources of error. First, our encoding methodology may not be flawless. In an ideal scenario, if an entity e is ideally represented by the string r , in practice, we may be working with another string r' that approximates r but is not identical. This type of error is what we call *misencoding*.



Figure 1.1: The Problem of Understanding

Second, our descriptions may not be perfect. Ideally, a description d should allow us to recreate the string r' (as the string r is typically unknown), but in practice, it generates another string r'' that is close to r' but not identical. This second type of error is what we denote as *inaccuracy*.

Finally, the third category of errors pertains to the descriptions themselves. Given humans' limited cognitive capabilities, we are interested in the shortest possible description d of e so that we can understand the entity, make predictions, or derive implications. However, our current best-known description d' is likely longer than d . We term this third type of error as *surfeit*.

In conclusion, we amalgamate these three types of errors - miscoding, inaccuracy, and surfeit - into a single metric known as *nescience*. This serves as a quantitative measure of our lack of knowledge about an entity.

This chapter provides an accessible introduction to our novel *theory of nescience*, accompanied by a concise review of its principal findings. While avoiding the use of complex mathematics, the chapter maintains a semi-formal description of concepts. To further elucidate the introduced notions, a variety of practical examples are presented throughout.

1.1 Entities

At the heart of our theory is the idea that there exists a non-empty collection of things, or *entities*, that scientists are trying to understand. This set of entities, which we'll refer to as \mathcal{E} , could be composed by anything that has been, is currently being, or can be studied in the future. And, we're assuming that it's possible for science to fully understand and explain at least some of these entities.

It's important to note that \mathcal{E} is not a well defined set, from a mathematical point of view, because the only condition is that it must be nonempty. This broad definition has both upsides and downsides. On one hand, it's a limitation because it means our definition of nescience (how much we don't know) isn't something we can easily calculate or measure, so we have to estimate it in real-world situations. On the other hand, it's an advantage because it means the new ideas and methods we're introducing in this book aren't just limited to solving specific problems in science - they can be used in all kinds of different fields.

In the theory of nescience, we don't allow universal sets - essentially, we can't posit the existence of a set \mathcal{E} that encompasses everything. The complication with universal sets is their violation of Cantor's theorem, as explained in Section 9.1. According to Cantor's theorem, the power set $\mathcal{P}(\mathcal{E})$ comprised of all possible subsets of \mathcal{E} has more elements than \mathcal{E} itself, contradicting the premise that \mathcal{E} encapsulates everything. Therefore, in the theory of nescience, \mathcal{E} must represent the set of "something".

Furthermore, not all conceivable sets are permitted. An instance of this is Russell's paradox, which postulates the set \mathcal{E} to consist of all sets that are not members of themselves. The paradox surfaces when we try to determine whether \mathcal{E} itself is a member of itself or not (as explained in Section 9.1). Thus, we require each set \mathcal{E} to be confined to a specific type of elements, precluding them from becoming members of themselves.

Typically, the set \mathcal{E} corresponds to a distinct field of knowledge, with its constituent elements varying based on how the theory is practically applied. Instances of such sets of entities might include: a collection of mathematical objects (abstract); the Animalia kingdom (living organisms); recognized and unrecognized human needs (abstract); all potential computer programs (strings of symbols), and so forth.

1.2 Representations

In many instances, entities cannot be directly scrutinized through scientific analysis, particularly if they are abstract. Consequently, we have to work with representations of those entities. We designate the collection of strings that encode the entities of \mathcal{E} as $\mathcal{R}_{\mathcal{E}}$. These strings, referred to as *representations*, may differ depending on the application of the theory of nescience. In certain scenarios, entities may inherently be string-based (e.g., computer programs), while others might be abstract objects that require encoding into string format (e.g., human needs). It's worth noting that a single entity $e \in \mathcal{E}$ could correspond to more than one valid representation in $\mathcal{R}_{\mathcal{E}}$. Transforming abstract entities into symbol strings in a manner that fully captures their complexities and intricacies remains a significant, unresolved challenge. As such, the exact composition of the set $\mathcal{R}_{\mathcal{E}}$ often remains unknown.

Ideally, we would like to establish an encoding function $f : \mathcal{E} \rightarrow \mathcal{R}_{\mathcal{E}}$ that maps a set of entities \mathcal{E} to the corresponding set of possible representations. However, in practice, defining such a function is challenging because the set \mathcal{E} is not a well defined set, meaning we can't conclusively determine what should or should not be included in this set. For instance, it will require further research to precisely define what constitutes a human need.

From a theoretical standpoint, the concept of an *oracle Turing machine* (see Chapter 4) could be employed to tackle this issue. A Turing machine is essentially a mathematical model of computation, whereas an oracle Turing machine can be likened to a computer model with internet access. We could presume that this computer could query a hypothetical external service, situated remotely, to verify whether a specific string r encodes *any* entity of \mathcal{E} . Notably, we cannot ask the oracle if the string r is a valid representation of the specific entity e we are interested in, as this would necessitate providing a valid string representation of e , which is generally unknown. The concept of the oracle machine, however, allows us to define a function from the collection of finite binary strings to the set of entities $f : \mathcal{B}^* \rightarrow \mathcal{E}$.

From a practical perspective, we typically approximate the set $\mathcal{R}_{\mathcal{E}}$ with another set $\hat{\mathcal{R}}_{\mathcal{E}} \subseteq \mathcal{B}^*$ of strings, which we consider to be adequate representations of the entities of \mathcal{E} . In scientific practice, these representations have traditionally taken the form of illustrations or images (e.g., in biology), collections of factual data (e.g., in sociology), or experimental results (e.g., in physics). With recent significant advancements in the capability of computers to gather and store data, a novel and potent method for encoding entities has emerged: using vast data sets as representations. It's essential to note that in the encoding process, our objective is not to find the shortest possible representation of the entities but to seek out high-quality representations.

It's crucial to acknowledge that in numerous practical scenarios, the chosen representations of abstract entities may not fully encapsulate all nuances of the original objects. This means we are grappling with oversimplified abstractions of reality, which could potentially curtail our capacity to make sweeping assertions about nature (see Chapter 10).

■ **Example 1.1** If we're studying animals (the set \mathcal{E}), we could use a binary encoding of their DNA (the set $\mathcal{R}_{\mathcal{E}}$) as representations. While our current technology doesn't allow us to bring a creature to life solely from its DNA, theoretically, it could be feasible. However, DNA alone doesn't fully replicate the original animal, as it doesn't include life experiences. For instance, how would we represent a cat that only has three legs due to an accident? That detail is not recorded in its DNA. If

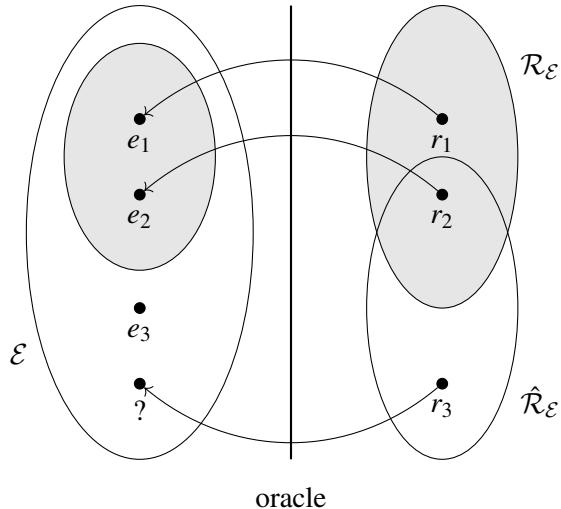


Figure 1.2: Entities and Representations.

our goal is to study the traits of certain species, working with the DNA of a representative sample of individuals within each species would be adequate. However, if we're studying specific individuals within a species, we would also need a way to encode each animal's history or the details not encapsulated by the DNA. ■

Working with string representations (the set \mathcal{R}_E) can lead to certain entities not being encoded by any representation (see the gray areas in Figure 1.2, specifically, entity e_3 lacks an encoding). For instance, if our set of entities is the set of real numbers, some numbers lack a representation, since we don't allow infinite strings as representations. Intuitively, this suggests that in many knowledge domains, the quantity of problems may exceed the number of solutions. Using approximations of representations (the set $\hat{\mathcal{R}}_E$) can also lead to some representations encoding incorrect entities (as in the case of representation r_3 in Figure 1.2). This is due to our incomplete knowledge possibly leading us to utilize incorrect representations for the entities of E . Another issue with incomplete knowledge is the possibility of 'unknown unknowns' - entities whose existence we're unaware of. For instance, representation r_1 in Figure 1.2 is not part of the set $\hat{\mathcal{R}}_E$ and is therefore overlooked by researchers despite its representation of a valid entity e_1 . Our objective includes exploring a procedure to uncover new, previously unknown, research entities from the set \mathcal{R}_E (refer to Section 1.11 and Chapter 15).

A *research area* A is a subset of topics, in other words, $A \subset \mathcal{R}_E$. In a practical context, areas prove valuable when their included topics share a common attribute. Our interest may lie in investigating our knowledge gaps about a particular area. For instance, if the set of topics is "animals", areas could include "invertebrates", "mammals", "birds", "amphibians", "reptiles", and "fish". To gauge our lack of knowledge about an area, we first need to describe the topics encompassed by it. However, as our understanding of the topics constituting an area is generally limited, we can only provide partial descriptions. Furthermore, as our comprehension of a research area evolves, the quantity of topics included in its known subset adjusts accordingly. As a result, we can only examine the properties of areas relative to our present knowledge.

■ **Example 1.2** Should our topic set revolve around "astronomy," a conceivable research area might be "habitable planets," a category wherein we have knowledge of only a few currently. A model for this area could be a description compiled from our existing knowledge of these habitable planets. ■



Figure 1.3: Entities, representations and descriptions.

1.3 Descriptions

Upon identifying the set \mathcal{R} of potential representations, our next task is to devise an appropriate method to describe them. In other words, we need to form theories or models that reflect our understanding of the workings of the world. Our human cognitive abilities are, by nature, limited. Therefore, we often depend on simplified models of nature to interpret observed phenomena and predict the outcomes of our actions.

Defining a valid description for an entity presents a challenging, as yet unresolved, issue. Take for example, the Berry paradox, which suggests the description, "the smallest positive integer not definable in less than twelve words". This statement itself becomes paradoxical as it essentially describes the number in only eleven words. To sidestep such paradoxes, the theory of nescience necessitates that a description for an entity should be a finite symbol string from which we can effectively and completely reconstruct one of the possible representations of the original entity. By "effectively reconstruct", we mean that our models should be computer programs that print the selected representation upon execution. From the era of Newton, science has focused on mathematical models, essentially pursuing a function set that thoroughly describes natural objects and their interrelations. The theory of nescience extends this concept, mandating that such models should be computable.

Descriptions are typically divided into two components: a Turing machine TM (a computer program) that encapsulates all the regularities found in the entity's representation (the compressible part), and an input string a that contains a literal description of the remaining elements (the non-compressible part). Formally speaking, a description for a representation $r \in \mathcal{R}$ is a string $\langle TM, a \rangle$ where $TM(a) = r$. This dualistic nature of descriptions parallels traditional distinctions such as theories and assumptions, theories and initial conditions, problems and specific problem instances, species and individuals, and so on. For instance, a description might consist of a system-modeling set of differential equations (the compressible part), accompanied by a compilation of initial conditions (the non-compressible part). The precise interpretation of the pair $\langle TM, a \rangle$ relies on the specific characteristics of the entity set to which the theory is applied.

Figure 1.3 visually illustrates the relationship between entities, representations, and descriptions. the set of all potential descriptions is signified by \mathcal{D} . Not all possible strings qualify as descriptions, as we insist that descriptions must be rooted in Turing machines, and not all possible descriptions outline valid representations.

■ Example 1.3 To offer an intuitive perspective, consider studying the macroscopic physical aspects of our world. The set of potential descriptions would incorporate elements such as Aristotelian physics, Cartesian physics, Newtonian physics, Einstein's theory of general relativity, and superstring theory, among others. At present, the best description would be Einstein's theory of general relativity, as the superstring theory is yet to be experimentally verified. ■

A representation $r \in \mathcal{R}$ can have several descriptions, and hence, the objective of science is to discover the shortest possible description, represented by d^* , that enables us to thoroughly recon-



Figure 1.4: Miscoding of topics.

struct the representation r . Unfortunately, no method or algorithm exists that, when given a string, returns the most concise computer program that outputs that string (refer to the incomputability result of Kolmogorov complexity in Chapter 6). In particular, no method exists that, when provided with a representation, discerns the shortest possible description. Consequently, science presents a non-computable problem, necessitating the compilation of heuristics that approximate the optimal solution. This compilation of heuristics is what we call a *scientific method*.

The theory of nescience concentrates on comprehending and quantitatively assessing the possible errors in the ideal process depicted in Figure 1.3. In Sections 1.4, 1.5, and 1.6, we propose a series of metrics to measure each potential source of error. In Section 1.7, we explore how to merge all these concepts into a single quantity, called *nescience*. This new nescience metric allows us to quantitatively determine the extent of our ignorance about a research entity.

1.4 Miscoding

As we've observed, in many scientific disciplines, the set \mathcal{E} of entities under examination might consist of abstract elements or various types of objects that cannot be easily represented as a string of symbols. There may also be instances where our comprehension of the elements in \mathcal{E} is limited, preventing us from encoding them appropriately. In such situations, we must rely on approximate representations rather than accurate ones. We aim to quantify the error introduced by the use of these inaccurate representations.

We propose to measure the *misCoding* of an inaccurate representation r' as the length of the shortest computer program capable of printing the accurate representation r when given r' as an input, denoted as $K(r|r')$. This is demonstrated in Figure 1.4, where \mathcal{R}_e represents the set of strings that accurately represent the entity e . Essentially, miscoding assesses the effort (determined by the length of a program, not the time it takes for the program to execute its task) needed to rectify the incorrect representation. If our representation includes errors, the program should identify and correct them. If our representation omits vital information necessary for fully encoding the entity, the program will include, hard-wired, this missing information.

However, this method does not wholly encapsulate our intuitive concept of miscoding. A problem occurs when our representation r' includes surplus information that is not required to encode the entity e . In such a case, our descriptions might include elements that model these unneeded symbols, resulting in artificially elongated descriptions. For instance, in an experiment, we might be measuring features that have no impact on the experiment's outcome. Given our limited understanding of the original entity, our current description might incorporate these features

as predictors. To circumvent this issue, we also need to calculate the length of the shortest computer program capable of printing the incorrect description r' when given the correct one r , i.e., $K(r'|r)$. Miscoding can then be defined as the maximum of these two lengths, $\max\{K(r|r'), K(r'|r)\}$.

Nevertheless, this latest definition still poses practical challenges. Most entities have multiple valid representations. Perhaps r' is a significant deviation from representation r_1 , but it is much closer to another, r_2 . It would be unjust to deem a description d that perfectly models r_2 as poor simply because it doesn't model r_1 , especially when both r_1 and r_2 represent the same entity e . One potential solution to this issue could involve examining $\min_{r \in \mathcal{R}_e} \{\max\{K(r|r'), K(r'|r)\}\}$.

■ **Example 1.4** Consider e as the abstract entity known as the "Pi constant", which is the ratio of a circle's circumference to its diameter. Let r represent the Wallis' product $2(\frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \cdot \dots)$, which is a valid representation of e . Suppose d is the description $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$. It wouldn't be fair to claim that d is an exceptionally poor description of the entity e just because it does not reproduce r . In reality, d generates the Leibniz's series $4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots)$, another valid representation of Pi. Even if Leibniz's series were unknown by mathematicians, it should not be classified as a miscoded representation of Pi. ■

As example 1.4 indicates, defining miscoding poses challenges because the set \mathcal{R}_e of valid representations for the entity e is generally unknown. Theoretically, we could rely again on the oracle Turing machine to solve this problem. However, as observed, we can't ask the oracle if the string r is a valid representation of our interested entity e (the set \mathcal{R}_e), since that would require us to provide a valid representation of e as a string of symbols, which typically can't be done. The only thing we can do is to ask the oracle how far the string r is from being a valid representation of any entity from the set of entities $\mathcal{R}_{\mathcal{E}}$.

Bearing in mind the aforementioned considerations, we define the miscoding of a representation r , denoted by $\mu(r)$, as follows:

$$\mu(r) = \min_{r_e \in \mathcal{R}_{\mathcal{E}}}^o \frac{\max\{K(r_e|r), K(r|r_e)\}}{\max\{K(r_e), K(r)\}}$$

where \min^o implies that the minimum function has to be computed by an oracle. Notice in the definition, we've introduced the normalization factor $K(r_e)$ —the length of the shortest program that can print r_e —since we're interested in comparing the miscoding of various, possibly unrelated, representations.

Given that miscoding doesn't utilize the original entity e as an argument, it's possible that what we're representing is different from what we anticipated. In other words, our representations and descriptions might be investigating an entirely different entity. This occurrence is quite common in scientific research practice.

■ **Example 1.5** In the late eighteenth century, chemist Joseph Priestley believed he was studying the non-existent entity named "phlogiston", which was thought to be a fire-like element contained within combustible bodies and released during combustion. However, he was actually studying a completely different entity known as "oxygen". ■

According to the theory of nescience, our role as researchers isn't solely to identify the appropriate representations of the entities within \mathcal{E} , but also to understand how this ideal oracle machine, which knows how to encode entities correctly, functions. That is, we need to comprehend why our representations serve as effective encodings of the entities being studied.

1.5 Inaccuracy

In the preceding section, we established a way to measure our lack of knowledge about a certain entity e . This was done in terms of miscoding, which results from utilizing an incorrect representa-

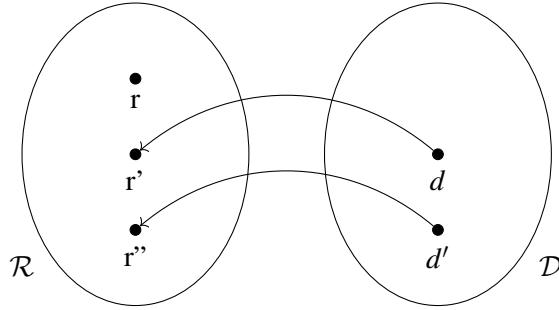


Figure 1.5: Inaccuracy of a description.

tion r' instead of the correct one, r . In this section, we aim to examine the extent of our lack of knowledge based on a given description.

In an ideal situation, we would use a description d that would allow us to completely rebuild the representation r' (keeping in mind that the true representation r might be unknown). However, in most real-world situations, this is not feasible. Usually, we work with a description d' that generates a string r'' . This string r'' is expected to be similar to the string r' , albeit not identical. In such instances, we say that the description d' is an *inaccurate* depiction of the representation r' (refer to Figure 1.5). Recall that the descriptions should be computer programs.

If a description is inaccurate for a representation, it is useful to have a quantitative measure of how much we deviate from accurately modeling the representation. A viable method to define this measure could be calculating the effort needed to rectify the output of our inaccurate description. In this context, the inaccuracy could be determined by the length of the shortest computer program that can generate the correct representation when fed with the incorrect one produced by the description. However, similar to the case with miscoding, to have a holistic understanding of the error associated with the description d , we must also calculate the difficulty of generating the inaccurate representation given the correct one. It's possible that our description d models elements unrelated to the representation r' , and merely ignoring these elements won't solve the problem.

Let $r' \in \mathcal{R}$ be a representation, and $d \in \mathcal{D}$ a description that produces the string r'' . We define the *inaccuracy* of the description d for the representation r' , symbolized as $\iota(d, r')$, by the formula:

$$\iota(d, r') = \frac{\max\{K(r'' | r'), K(r' | r'')\}}{\max\{K(r'), K(r'')\}}$$

Again, in order to have the most comprehensive interpretation of the concept of inaccuracy, we divide by the normalization factor $\max\{K(r'), K(r'')\}$. This allows us to compare multiple descriptions for the same representation, as well as descriptions for different representations.

We favor the term *inaccuracy* over the term *error*. An error comprises two components: precision and accuracy. However, precision typically pertains to continuous systems. Since we're dealing with discrete strings of symbols in this context, it's not quite appropriate to discuss precision.

In practice, calculating the inaccuracy associated with the description of a representation is a challenging task. As previously mentioned, determining the length of the shortest computer program that can print a string is a non-computable problem. If the original entities are texts themselves, we could approximate the inaccuracy using compression algorithms. Here, the Kolmogorov complexity is approximated by the length of the compressed text using a compressor. If the topics are abstract entities, such as mathematical concepts, their descriptions could be derived from the result of an experiment. Hence, the inaccuracy could be based on the model's error (for instance, by calculating the length of additional information required to thoroughly describe the experiment's results given the model). In this regard, our definition of inaccuracy is a generalization of the concept of error. It can be applied to various types of entities, not only those that can be encoded as datasets.

■ **Example 1.6** The topic of Newton's second law of motion $F = ma$ could be encoded using the results of an experiment conducted with objects of varying masses to which different forces are applied, and then measuring the acceleration. The dataset would comprise the masses, forces, and accelerations achieved for each combination of mass and force. However, if we're interested in acceleration due to gravity, forces and masses cancel each other out, leaving acceleration as the only value to measure. If the experiment results in a large collection of measurements, the encoding of the entity would be quite lengthy. However, with entity encodings, we're not striving to find the shortest possible strings. Instead, we aim for comprehensive and accurate encodings of topics. For this example, we'll use the results of an experiment performed by the National Bureau of Standards in Washington D.C. between May 1934 and July 1935. The dataset consists of 81 measurements, each expressed in centimeters per second squared, for example, 980,078. In practice, we approximate the quantity $\frac{K(t|m)}{K(t)}$ by $\frac{C(D|M)}{l(D)}$, where $C(D | M)$ is the length of the compressed version of the data given the model (using a standard compressor), and $C(D)$ is the length of the compressed data. To encode the dataset, we require 20 bits per measure (using a uniform code, as explained in Chapter 5), making the full dataset encoding require 1,620 bits. Assuming that our model predicts gravity of $980,000\text{cm/s}^2$ plus a random error following a normal distribution (estimated using a maximum likelihood approach), it would take 453 bits to encode the data given the model (refer to Chapter 17 for more information on encoding a dataset given a model). Therefore, the model's approximated inaccuracy is $\frac{453}{1,620} = 0.27$. ■

As it was pointed out by Example 1.6, when dealing with representations based on experiments, we have to take into account that there is no way, from a logical point of view, to determine which one is the factor that contributes the more to our unknown, a wrong model (inaccuracy) or a wrong experiment (misencoding).

1.6 Surfeit

If a concept or process requires extensive time and effort to explain, it's likely that we do not fully comprehend it and our understanding remains incomplete. Lengthy explanations often comprise unnecessary components. In our view, a primary aim of science should be to minimize these superfluous elements as much as possible. We depend on descriptions, typically mathematical models, to predict future outcomes based on the past, understand the link between cause and effect, and to create machinery that resolves problems. There is a great importance placed on discovering the most concise models that describe how systems function, so they can be accommodated within our cognitive limitations and simplify our tasks as scientists and engineers¹.

The theoretical limit of our knowledge about a representation, denoted as its perfect description d^* , is given by the shortest possible computer program that can reconstruct that representation. The excess or surfeit of a description d' can be calculated by comparing the length of this specific description to the length of the best possible description d^* for that representation (refer to Figure 1.6).

Given an entity e , and one of its representations r , we define the *surfeit* of the description d for the representation r , denoted by $\sigma(d, r)$, by:

$$\sigma(d, r) = \frac{l(d) - K(r)}{l(d)}$$

where $l(d)$ is the length (number of symbols) of the description d , and $K(r)$ is the length of the shortest possible description for r , that is, d^* . Since we are interested not only in measuring

¹In a (hopefully) not too distant future, when all scientific reasoning is conducted by computers, the need for the shortest possible models will no longer be a priority, and it will be relegated to a mere scientific curiosity with limited practical significance.



Figure 1.6: Surfeit of a model.

the surfeit of the description of individual representations, but also in comparing the surfeit of representations for multiple, potentially unrelated, entities, we have to divide the difference $l(d) - K(r)$ by the normalization factor $l(d)$. Unfortunately, in general, we do not know the shortest description of a representation, since our knowledge is incomplete, and so, surfeit is a quantity that has to be approximated in practice.

For a given entity e , and one of its representations r , we define the *surfeit* of the description d for the representation r , symbolized by $\sigma(d, r)$, as:

$$\sigma(d, r) = \frac{l(d) - K(r)}{l(d)}$$

where $l(d)$ represents the length (number of symbols) of the description d , and $K(r)$ signifies the length of the shortest possible description for r , that is, d^* . As our interest extends beyond measuring the surfeit of the description of individual representations and includes comparing the surfeit of representations for several, potentially unrelated, entities, we must divide the difference $l(d) - K(r)$ by the normalization factor $l(d)$. Regrettably, due to our incomplete knowledge, we generally do not know the shortest description of a representation, hence, surfeit is a value that must be estimated in practice.

If we were able to come up with a perfect description for a representation, that string of symbols must be incompressible, otherwise it would contain superfluous elements that can be removed, and so, either it is not incompressible or it is not perfect. Given that an incompressible string is a random sequence of symbols (see Section 6.6), an important consequence of our theory is that perfect knowledge implies randomness. The common understanding is that it is not possible to make sense from something that is random, since this is what randomness is all about. However, in the theory of nescience, by random description we mean a model that contains the maximum amount of information in the smallest space possible (it contains no redundant elements). Please mind that the converse does not hold; that is, a random description does not imply perfect knowledge. It might be possible we keep improving a theory until it becomes random, but later on we find a new and shorter theory (probably based on another encoding) that describes the same entity, and this new theory is not random yet.

Should we be able to formulate a perfect description for a representation, this sequence of symbols should be incompressible; otherwise, it would contain unnecessary elements that can be eliminated, thus, it would either be non-incompressible or imperfect. Considering that an incompressible string is a random sequence of symbols (refer to Section 6.6), a significant inference from our theory is that perfect knowledge entails randomness. The conventional belief is that deriving meaningful insights from something random is unfeasible, as randomness inherently lacks order or predictability. However, in the theory of nescience, a random description signifies a model that encapsulates the maximum amount of information in the smallest possible space. Please note that the reverse is not true; a random description does not automatically denote perfect knowledge.

Kowledge Area	Redundancy
Mathematics	0.351
Computer Science	0.443
Chemistry	0.466
Biology	0.475
Psychology	0.528
Epistemology	0.530
Sociology	0.543

Table 1.1: Redundancy of Scientific Areas

It might be feasible to refine a theory until it becomes random, but later we may discover a shorter, new theory (probably based on a different encoding) that describes the same entity, and this new theory has not yet reached randomness.

Randomness sets a boundary on the extent of our knowledge about a specific entity. Instead of posing an obstacle, the proper comprehension of this absolute epistemological limit provides an opportunity to deepen our understanding of unresolved problems. Furthermore, since the constraint lies in enhancing our knowledge about already identified research entities rather than discovering new ones (which are presumed to be infinite), our understanding of randomness can also be utilized to discover new research entities.

With our definition of surfeit, in which lengthier explanations are deemed inferior, we aren't implying that textbooks should always strive for utmost brevity. Contrarily, in certain situations, we anticipate textbooks to be highly redundant. A concise book contains an abundance of information in a very condensed space, making it challenging for humans to assimilate (understand) that information. However, a redundant textbook (such as this one) presents the same amount of information but in a larger space, hence, its content is easier to comprehend. Moreover, in fields outside of science, redundancy may be desirable. For instance, in law, redundancy aids lawyers in memorizing legal texts, and in music, repetition can contribute positively to harmony, as exemplified in a canon.

The concept of surfeit can also be extended to encompass entire research areas, allowing us not only to calculate the surfeit of a specific area as a whole but also to compare the surfeit across multiple, unrelated areas.

■ **Example 1.7** By utilizing the concept of "category" from Wikipedia, which closely aligns with our concept of "research area", we can compute the surfeit of the major scientific areas (refer to Chapter 12 for more details on how to practically compute surfeit). For instance, as demonstrated in Table 1.1², in general, our understanding of Mathematics surpasses our understanding of Computer Science, since the redundancy of Mathematics is 0.351, and the redundancy of Computer Science is 0.443. The application of the concept of redundancy to other areas largely aligns with our intuitive notion of which knowledge areas are better understood. ■

1.7 Nescience

Nescience is an old fashioned English word meaning “lack of knowledge or awareness”. In this sense, it seems to mean exactly the same as the word “ignorance”, however there is a subtle difference: ignorance refers the lack of knowledge when knowledge is there (we do not know but we could learn, for example, by reading a book), and nescience refers to the lack of knowledge when knowledge is not there (we do not know, and it is not possible to know, since nobody knows).

²Data from October 2014.

The theory of nescience has been developed with the aim of quantitatively measuring how much we do not know when knowledge is not there, that is, to quantify how much we, as humankind, do not know.

Intuitively, how much we do not know about an entity has to be computed based on the miscoding, inaccuracy and surfeit of a representation and a description, since those metrics summarize all possible types of mistakes we can make. Miscoding because it tell us how well the representation encodes the original entity under study; inaccuracy because it says how well the description models the representation; and surfeit because it tell us how good is the description itself. The best combinations of representations and descriptions are those who present a low miscoding, low inaccuracy and low surfeit. Unfortunately, those quantities are conflicting, in the sense that if we reduce one of them, the others may increase. For example, if we use a more complex description usually the inaccuracy will decrease but the surfeit will increase.

A pair (d, r) composed by a description and a representation is Pareto optimal if there does not exist another pair (d', r') such that decreases at least one of the components of nescience, that is miscoding, inaccuracy and surfeit, without increasing another component. We are interested in a local version of the concept of Pareto optimality, since it might happen that the description d' is so far from the description d that it does not represent anymore the entity e encoded by r (given the insight of the oracle).

Pareto optimality allow us to find a family of candidate pairs (d, r) that provide good explanation of an entity e . However, in science we prefer to select a single description as the model of a research entity. In order to select that description we have to provide a utility function that allow us to classify and order the candidate descriptions. The form of this utility function is something that depends on the area in which the theory of nescience is being applied. For example, in case of entities encoded as datasets (machine learning) a good candidate utility function is the harmonic mean (see Chapter 17). The *harmonic nescience* of the representation r given the description d , denoted by $v_H(d, r)$, is defined as:

$$v_H(d, r) = \frac{3}{\mu(r)^{-1} + \iota(d, r)^{-1} + \sigma(d, r)^{-1}}$$

In the practice of science, we usually fix r and talk about the nescience of the representation r instead of the original entity e . If we have multiple candidate models d_1, d_2, \dots, d_n for the representation r , we say that our *current best description*, denoted by \hat{d} , is the description that present the lowest nescience given a representation r .

■ **Example 1.8 TODO: Review this example.** We are interested in understanding which are the factors that affect the price of houses. The encoding of that topic will be given by a dataset corresponding to 506 random samples taken from the suburbs of Boston. For each sample, 14 attributes have been measured, like for example the average number of rooms per dwelling, the per capita crime rate by town or the accessibility to radial highways. Our models will be based on a decision tree (a collection of nested if-else clauses), showing which are the most relevant factors that affect the price. The problem at hand is to identify the ideal depth of that tree, that is, how many if-else decision we should allow in the model that explain the behavior of prices. In order to do that, we compute the best trees from a depth level of 2 to a depth level of 8. In Table 1.2 left we have a plot of the mean squared error of the best model at each level, using different subsets for training and testing in order to avoid the overfitting of models. As we can see, the optimal level is 5, since increasing beyond this point means that there will be almost no further gain. In Table 1.2 right we can see the same study but applying our concept of nescience. In this latter case, we see that the optimal level is 3. That is, beyond that level it might be possible that our error decreases, but the model becomes so complex that it makes interpretation very difficult, and so, our understanding of

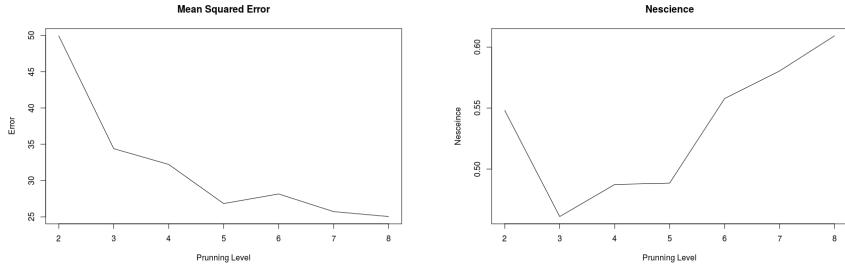


Table 1.2: RMSE and Nescience of Models

the topic does not increase³. ■

We are interested in comparing the nescience, i.e. how much we do not know, of different research entities. For example, mathematicians claim that we do not know almost anything about differential equations, and sociologists claim that we do not know almost anything about the causes of armed conflicts. As it is shown in Example 1.9, this "we do not know almost anything" is much higher in case of armed conflicts than differential equations.

■ **Example 1.9** We can compare how well we understand the "causes of the first world war" with how well we understand the "dynamics of populations", in this particular example, represented by the Lotka–Volterra differential equation. In order to compare in practice these research topics we need a description, as complete and accurate as possible, of our current understanding about them. In case of abstract objects, we could use as descriptions the ones contained in reference works, since what we need is an encyclopedic coverage. For this example, we have used the descriptions found in the corresponding pages of the Wikipedia online encyclopedia⁴,⁵. After a pre-processing of the original text, in which Wikimedia tags were removed, the files were compressed with the 7z compression utility (see Chapter 19 for a detailed description of this process). Given this approach we have estimated that the redundancy of the causes of the first world war is 0.67, and the redundancy of the dynamics of population is 0.59. Those numbers suggest that we know better how the dynamics of populations work than the causes of this particular armed conflict. Unfortunately, not only is redundancy an approximation of surfeit, but also, in order to fully characterize our unknown we have to take into account the error of both descriptions. Moreover, it might happen that these pages at Wikipedia are not the best current descriptions available for those topics. ■

When the nescience of a representation r is equal to zero ($v(d, r) = 0$), we say that we have reached a *perfect knowledge* about an entity. In practice it is not possible to know when we have reached the shortest possible description of the best possible valid representation, that is, when we have found the ultimate theory, since representations require to query the oracle, and descriptions are based on the uncomputable Kolmogorov complexity.

1.8 Evolution of Nescience

In this book we assume that the final objective of science is to discover those descriptions and representations with the lowest possible nescience. The general approach is to produce a series of candidate descriptions and representations over time, each one with a lower nescience than the previous one, until perfect knowledge is reached. New descriptions could be based on novel theories that explain the entity, refinements over already existing ones, reducing the number of

³Please mind that nescience and interpretability are not equivalent concepts, in the sense that we can find models with very low nescience that it are beyond the human capabilities of interpretation.

⁴https://en.wikipedia.org/wiki/Causes_of_World_War_I (retrieved on August 2017)

⁵https://en.wikipedia.org/wiki/Lotk-Volterra_equations (retrieved on August 2017)

assumptions, etc. Nescience can also decrease by means of discovering better representations of the entities under study.

If performed properly, the nescience of an entity should be strictly decreasing as new descriptions and representations appear, since we should not accept a new description or representation as an improvement over the existing ones unless it reduces the nescience. It might happen that a new description presents a higher surfeit, or a higher inaccuracy, but never both things at the same time, since that would imply a higher nescience. Of course, in practice things are not that easy, since our current values of miscoding, inaccuracy and surfeit as just estimations of the real values, and they could be wrong estimations. For a practical point of view, we only require that nescience decrease over time on average.

■ Example 1.10 The concept of nescience can be applied to measure how well we understand current computer software, like for example operating systems, databases, or productivity applications. The surfeit of the software could be based on the compressibility of the source code, and the inaccuracy on the number of bugs found. In Figure 1.7 we can see the evolution of the nescience in the latest published versions of the open source database SQLite⁶. As we can observe (given the regression line) the nescience of SQLite decreases with time, and so, we can conclude that as new versions appear this application better solves the problem at hand. In Chapter 18 we will see that this is not the case for most of the existing software platforms and applications. ■



Figure 1.7: Evolution of Nescience

We can use this property of reduction of nescience as a characterization of what constitutes a valid scientific discipline and what is not (what it is called the demarcation problem in philosophy of science). In case of non-scientific theories (pseudosciences and others) nescience does not decrease, on average, when new representations or new descriptions are available. That is, in pseudosciences we do not learn anything new when we do further research.

■ Example 1.11 A trading robot is a computer program that gets as input real time quotes from the stock markets, or foreign currency exchanges, and based on an algorithm decides how to invest money, in general by means of opening very short-time positions (what it is called intra-day trading). Many of these robots are based on technical indicators, that is, metrics that are derived from current and past prices (like moving averages, or resistance levels) to forecast future prices. It is an open question if it is possible to make any money using those robots, that is, if they have a positive mathematical expectancy sufficiently high to cover brokers commissions. We can study this problem with the aid of the theory of nescience. In order to do that, we have randomly selected 40 trading robots developed over a period of 6 years, and we have tested them (see Section 18.3 for

⁶www.sqlite.org

more information). In Figure 1.8 we can see the evolution of the nescience of these robots along time. As we can observe, nescience does not decrease, in fact it increases. ■

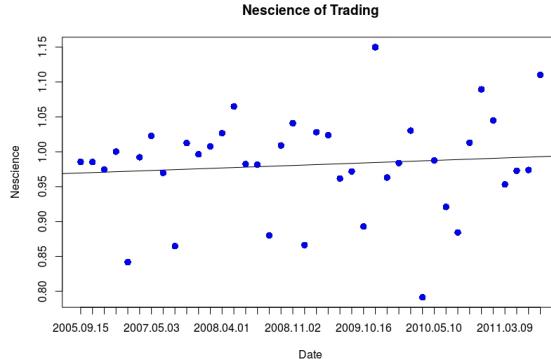


Figure 1.8: Nescience of Forex Trading Robots

1.9 Other Metrics

Nescience is a metric that can be used to identify the most interesting topics from the point of view of research. Since nescience is a measure of our ignorance about a topic, the higher the nescience the more opportunities for research. In this section we are going to propose other metrics that can complement nescience in the task of measuring the interest of research topics. These metrics will be helpful, not only for the classification of individual research topics, but also for the development of a methodology for the discovery of potential solutions to open problems (Section 1.10), and the discovery of new research topics (Section 1.11).

One of these new metrics for the classification of research topics is *relevance*. Relevance is a measure of the impact that a topic has on people's life. Intuitively, the higher the relevance of a topic, the higher its potential as a source of interesting problems, since we will be working on problems that affect many people directly. In order to measure the impact of a research topic we have to construct what we call the relevance graph, a figure that describes how people are affected by research topics (see Figure 1.9).

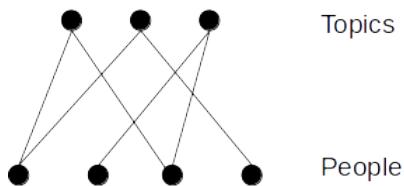


Figure 1.9: Relevance Graph

The relevance graph connects all known topics and all possible human beings. A link between a topic and a person means that this person is affected by the topic, not that he is interested in this particular topic. For example, somebody that is trying to find a cure to diabetes will not be connected to the research topic diabetes, however, somebody that actually suffer from diabetes will be. The higher the relevance of a topic, the higher its potential as a source of interesting problems to solve. In this sense, the research topic "how to cure diabetes" is more relevant than the research topic "how far dog fleas can jump", since more people are affected by the former than by the latter. The exact meaning of "*being affected by*" is an abstract concept that has to be approximated in

practice. For example, we could claim that the wife of a man that has diabetes is somehow affected by the disease as well. In Part III of this book we will see some examples of how to approximate this abstract quantity.

Given these two metrics, nescience and relevance, we can provide a quantitative measure of the interestingness of a topic as a source of interesting problems, that is, how likely is that the topic can be used as part of a new interesting research project. We define this quantity as a function of the relevance and nescience of the topic (for example, the normalized product of both quantities). Intuitively, a topic is interesting as a source of new problems if it has a large relevance (it has high impact in people's life) and a large nescience (it is not very well understood). In Figure 1.10 is graphically depicted the idea. For example, the Pythagoras' theorem has some relevance, since people life's can be indirectly affected by its implications, but since it is a very well understood theorem (our nescience is very low), it is not a very interesting research topic by itself. The World War I is very relevant, because it had a huge impact on many people's life, and also it is not very well understood topic as we have seen in Example 1.9. So, according to our definition, it has a huge potential as a source of new interesting research problems.



Figure 1.10: Interestingness of Topics

■ Example 1.12 We can use the collection of Wikipedia articles to identify topics that are interesting as a source of new problems. The starting point of our analysis is the classification contained in the scientific disciplines category of Wikipedia. This category is organized⁷ into the following main areas: "applied sciences", "behavioral sciences", "cognitive sciences", "formal sciences", "natural sciences", "physical sciences", and "social sciences". In order to evaluate the classification metrics proposed we have used the set of topics corresponding to all pages under any of the subcategories contained in the category "theory of computation" (a subcategory of the category "theoretical computer science", that belongs to the area "formal sciences"). Pages were cleaned up using the same procedure described in Example 1.9, and the relevance was estimated based on the number of unique visits to each page (please, refer to Chapter 19 for more information about this process). Topics that fit our intuitive idea of problem, that is, not very well understood concepts with a high relevance, could include "arithmetical hierarchy" (0.72), "halting problem" (0.65), "floating point" (0.61), "quantum computer" (0.57), and "computable function" (0.55). ■

The Pythagoras' theorem is not a very interesting research topic by itself, however, it is a very important theorem, since it can be applied to solve many practical problems. A new metric is required to capture this concept of topic that is important because it can be used as a tool to solve other problems. We define the *maturity* of a topic as the inverse of nescience. Intuitively, the more mature a topic is the higher its potential applicability as a tool to solve other open problems, since we know very well how the topic works and how it can be successfully applied. In general, highly immature topics should not be applied to solve open problems, since they could provide wrong

⁷Data from November 2014.

answers.

Besides maturity, we also introduce the metric of *applicability* of a topic. Applicability is based on the concept of *applicability graph*. An applicability graph is a directed graph between the research topics (see Figure 1.11). An arrow between two topics means that the first topic been successfully applied to explain or solve the second topic. For example, the topic “graph theory” has been applied to the topic “recommendation engines”, since graph theory has been used to solve the problem of which products we should advertise to potential customers on Internet. Given this graph, we define the applicability of a topic as the number of problems to which the topic has been successfully applied. The higher the applicability of a topic, the higher its potential as a tool that can be applied to solve new problems.



Figure 1.11: Applicability Graph

Finally, we define the concept of interestingness of a topic as a source of interesting tools, that is, how likely is that the topic can be used to solve a new problem, as a function of its maturity and applicability. Intuitively, a topic is interesting as a tool if it has been already applied to many other problems, and it is very well understood topic. For example, the Pythagoras’ theorem, although not very relevant as a source of interesting problems, it is very relevant as a source of new applications to other open problems.

■ **Example 1.13** We can use the collection of Wikipedia scientific articles to identify topics with high potential as tools. The procedure used in this example is similar to the one described in Example 1.12, except that the metric applicability has been estimated based in the collection of internal links within Wikipedia itself. The rationale is that if a page is referenced many times by other Wikipedia pages, it is highly likely that it contains useful information. Using this procedure, some examples of topics with high interest as tools in the area of theoretical computer science include "recursion" (0.42), "state space" (0.42), "abstract machine" (0.41), or "ternary numeral system" (0.48). ■

1.10 Interesting Research Questions

In the theory of nescience we distinguish two kinds of unknowns, the *known unknown* and the *unknown unknown*. By known unknown we mean all those already known problems for which we do not know their solutions, for example, nobody knows how to cure diabetes, but we know what diabetes is and we are aware that nobody knows how to cure it. By unknown unknown we mean the collection of unknown problems, that is, all those problems that have not been found yet, like for example the Eldermeyer’s disease⁸. The area composed by the unknown unknown problems is a highly interesting one, since it contains those research topics that will be addressed in the future. One of the main goals of this book is to help scientists discover the topics that lay in this unknown unknown area, since that would bring to the present the research problems of the future (see Section 1.11). In this section we focus on how to solve the problems of the known unknown area.

⁸We cannot say anything more about the Eldermeyer’s disease, since Dra. Eldermeyer will born next year, and it will take her 34 years more to discover the disease named after her.

An *interesting question* is a ordered pair of topics t and p , where t has a high interestingness as tool, and p has high interestingness as problem. Intuitively, the questions would be something like “can we apply the tool described by topic t to solve the problem described by topic p ?” The interestingness of the new questions will be measured by means of a function of the interestingness of t and p themselves. In practice what we have to do is to compute all the possible combinations of tools and questions and select those with higher combined interestingness (see Figure 1.12).

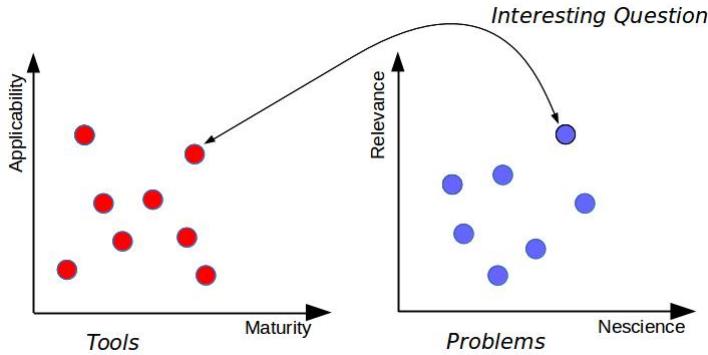


Figure 1.12: Interesting Questions

An interesting question is *intradisciplinary* if it combines two topics that are studied in the framework of the same research area (e.g., computer science). An interesting question is *interdisciplinary* if it combines two topics of different research areas (e.g., computer science and philosophy). In principle, the most innovative questions would be interdisciplinary questions, because the probability that somebody has thought about them is lower, since it requires specialists in both research areas working together to come up with that particular question.

■ Example 1.14 We could combine the topics with high interestingness as tools found in the area of "computer science" with those topics with high interestingness as problems found in the area of "biochemistry" in order to find new interesting interdisciplinary questions. Some examples of the kind of questions we can find with this approach include: "can we use regular expressions to identify DNA genes?" or "can we use a recursive algorithm to characterize proteins tertiary structure?" ■

Once we have identified an interesting research question, we could use the concept of *conditional model* to see if the tool can help us to understand, or solve, the open problem. A conditional model of a topic t given a perfect model of a second topic s is also a string in the form $\langle TM, a \rangle$, but in this case we require that $TM(\langle m_s^*, a \rangle) = t$. That is, the Turin machine TM is able to print t when it has as input both, the incompressible part a , and a perfect model m_s^* for the topic s . If the conditional complexity of the open problem given the tool is smaller than its original complexity, then the tool is helpful to solve the problem. The more the tool reduce the length of the conditional model with respect to the original model, the better.

Please note that the methodology presented here is a generic one, in the sense that it can be applied to multiple domains, not only to the discovery of new interesting research questions. The metrics and methods described can be applied to any area where there is a large collection of interrelated describable objects and we are interested in discovering new, previously unconsidered, objects. The exact definition of concepts like relevance graph, applicability graph or maturity will depend on the area in which the methodology is being applied. In Part III of this book, we will describe some examples of other applications, for example, to the identification of new software quality tests.

1.11 New Research Entities

We have mentioned in the previous section the existence of an *unknown unknown* area composed by those problems that not only we do not know how to solve them, but also we are not even aware they exist. We are interested in providing a procedure to discover new research entities located in this important area. A possible approach could be by randomly selecting a binary string and asking to the oracle if that string is close to the representation of a (hopefully unknown) entity. However, given the huge amount of candidate strings, that procedure is not feasible in practice.

In this book we will investigate an alternative approach, based on the combination of already known topics (see Chapter 15), what we call *joint topics*. If $r_1, r_2 \in \mathcal{R}$ are two different representations of two different entities, the joint topic of r_1 and r_2 is the concatenated string r_1r_2 (see Figure 1.13)⁹. For example, if $r_1 \in \mathcal{R}$ is a representation of the entity "maximum entropy", and $r_2 \in \mathcal{R}$ a representation of the entity "probability distribution", the entity represented by the joint topic r_1r_2 would be "maximum entropy probability distribution".



Figure 1.13: Discovering new research entities.

A *new research topic* is a topic that lies in the unknown unknown area. Since the surfeit (resp. inaccuracy) of joining two topics is higher than the superfeit (resp. inaccuracy) of any of them isolated, we can identify new interesting research topics by combining already existing interesting problems (see Figure 1.14). Formally, a new research topic is unordered pair of topics r_1 and r_2 , where both r_1 and r_2 have a high interestingness as problems. In practice, we have to compute all the possible combinations of those problems with very large interestingness with themselves, and select the ones with the higher potential. The exact meaning of the new topic that results by merging existing problems is something that has to be discovered with further research.

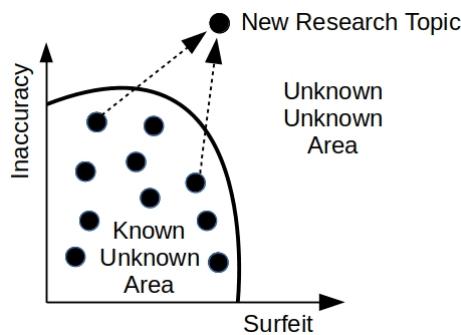


Figure 1.14: New Research Topics

⁹As we will see in Chapter 9, we require \mathcal{R} to be closed under the operation of concatenation of representations, that is, for any two representations $r, s \in \mathcal{R}$ we have that $rs \in \mathcal{R}$.

■ **Example 1.15** We could combine the most interesting topics of the area of "theoretical computer science" with the most interesting topics of the area "phenomenology" in order to identify the most promising combinations. For example, by combining "minimum complexity computer programs" and "self awareness" we obtain that a potential new research topic could be "minimum complexity self-aware computers"; that is, investigating the minimum complexity required for a computer program to have the capacity of being self-aware. ■

We can extend our method to find new research topics with additional metrics. For example, by means of adding the relevance of topics, we could discover new interesting research topics that are also relevant.

1.12 References and Further Reading

The idea that perfect descriptions must be random has been already mentioned by other authors (see for example [[mosterin2016conceptos](#)]). The gravity dataset used in Example 1.6 comes from [[cressie1982playing](#)], and a more detailed analysis can be found at [[davison1997bootstrap](#)]. The Boston housing dataset referred in Example 1.8 was published in [[harrison1978hedonic](#)] and further discussed in [[belsley2005regression](#)]; for more information about decision trees see for example [[james2013introduction](#)]. For more information about trading systems and technical indicators see for example [[kaufman2013trading](#)], and how to quantitatively test if a system is profitable to [[pardo1992design](#)]. The Lotka-Voterra model for population dynamics was originally described in [[lotka1920analytical](#)]. How to apply graph analysis to recommendation engines is covered in [[cordobes2015empirical](#)]. And finally, if the reader is interested in knowing how far a dog flea can jump, please refer to [[cadiergues2000comparison](#)].

The discipline of epistemology is more oriented to understand what we do know as individuals, meanwhile in this book we are interested in what we know as humankind; moreover, the kind of knowledge addressed by epistemology is not necessarily the scientific knowledge subject of the theory of nescience. Perhaps, the area of epistemology more interesting for our purposes is what the epistemologists call knowing by testimony. A good, and easy to read, introduction to the discipline of epistemology is [[nagel2014knowledge](#)].

Part 1: Background

2 Discrete Mathematics 43

- 2.1 Sets, Relations and Functions
- 2.2 Strings and Languages
- 2.3 Counting Methods
- 2.4 Matrices
- 2.5 Graphs

3 Discrete Probability 55

- 3.1 Foundations
- 3.2 Conditional Probability
- 3.3 Random Variables
- 3.4 Random Samples
- 3.5 Characterizing Distributions
- 3.6 Common Distributions
- 3.7 Large Random Samples

4 Computability 77

- 4.1 Turing Machines
- 4.2 Universal Turing Machines
- 4.3 Non-Computable Problems
- 4.4 Computable Functions and Sets
- 4.5 Oracle Turing Machine
- 4.6 Computational Complexity

5 Coding 91

- 5.1 Coding
- 5.2 Kraft Inequality
- 5.3 Optimal Codes
- 5.4 Entropy
- 5.5 Huffman Algorithm
- 5.6 Discretization Algorithms

6 Complexity 109

- 6.1 Strings Complexity
- 6.2 Properties of Complexity
- 6.3 Joint Kolmogorov Complexity
- 6.4 Conditional Kolmogorov complexity
- 6.5 Information Distance
- 6.6 Incompressibility and Randomness

7 Learning 121

- 7.1 Statistical Inference
- 7.2 Machine Learning
- 7.3 Minimum Message Length
- 7.4 Minimum Description Length
- 7.5 Multiobjective Optimization

8 Philosophy of Science 143

- 8.1 Metaphysics
- 8.2 Scientific Representation
- 8.3 Models in Science
- 8.4 Scientific Theories
- 8.5 The Scientific Method
- 8.6 Scientific Discovery

2. Discrete Mathematics

*Mathematics may be defined as the subject in which
we never know what we are talking about,
nor whether what we are saying is true.*

Bertrand Russell

The majority of mathematical concepts employed throughout this book fall within the realm of *discrete mathematics*. This field of study focuses on mathematical objects that possess distinct or individual values, as opposed to continuous ones. This book utilizes a variety of discrete objects, including integers, strings, graphs, and computer programs. A defining characteristic of discrete sets is their countability, meaning they can be enumerated with natural numbers. In contrast, we will scarcely apply continuous mathematics, such as calculus, to the theoretical formulations within the theory of nescience.

Our primary interest in discrete mathematics stems from its relevance to computers. The theory of nescience draws upon various facets of computer science, such as algorithms, coding, and string complexity. Computers function in discrete stages and process data stored in discrete memory units. We are captivated by computers due to our ambition to apply our theoretical explorations to a wide range of real-world objects. We believe computers provide the most appropriate means of modeling our world. While pure mathematics often delves into abstract objects without concern for their representations, the theory of nescience places great importance on the representation (or encoding) of objects.

This chapter serves as a brief overview of the fundamental concepts of discrete mathematics, and as such, does not provide formal definitions or prove theorems. Discrete mathematics is a highly diverse and vast field of study. We will only focus on those components essential for understanding the theory of nescience. Certain involved theories (such as computability, information, complexity, and so forth) demand a more extensive coverage and are therefore explored in individual chapters. The References section includes a list of recommended books that delve into the topics addressed in this chapter in greater detail.

This chapter serves as a brief overview of the fundamental concepts of discrete mathematics,

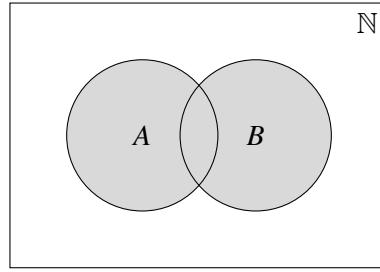


Figure 2.1: Representation of $A \cup B$ as a Venn Diagram

introducing topics such as sets, strings and languages, counting methods, matrices, and graphs. Though we do not provide formal definitions or prove theorems in this overview, these subjects offer a foundation for the theories and concepts discussed in subsequent sections. Discrete mathematics is a highly diverse and vast field of study. Our focus will be on those components essential for understanding the theory of nescience. Certain involved theories (such as computability, information, complexity, and so forth) demand a more extensive coverage and are therefore explored in individual chapters.

The References section includes a list of recommended books that delve into the topics addressed in this chapter in greater detail. By using this list, readers interested in deepening their understanding of these topics can explore each in more detail, complementing the fundamental overview presented in this chapter.

2.1 Sets, Relations and Functions

The sets of *natural*, *integers*, *rational*, and *real* numbers are represented by \mathbb{N} , \mathbb{Z} , \mathbb{Q} , and \mathbb{R} respectively, which includes the number 0. The *positive integers* are denoted by \mathbb{Z}^+ , and the *positive reals* by \mathbb{R}^+ , where both of these sets incorporate the number 0. Let A be a *set*. We signify that x is an *element* of A by the notation $x \in A$, and that x is not an element of A by $x \notin A$. Elements of a set can be enumerated using braces, such as $A = \{0, 1, 2, 3\}$, or they can be defined by conditions via the *set-builder* notation, for instance $A = \{x \in \mathbb{N} : x < 4\}$, with the stipulation that the *universe* of the set be clearly established.

Suppose A and B are two sets. We use the notation $A = B$ to denote that the sets are *equal*. The expression $A \subseteq B$ signifies that A is a *subset or equal* to B , and $A \subset B$ is used to indicate that A is a *proper subset* of B (implying A is not equivalent to B). The condition $A = B$ holds true if, and only if, both $A \subseteq B$ and $B \subseteq A$ are satisfied concurrently. The symbol \emptyset represents the *empty set*, a set that contains no elements.

■ **Example 2.1** For every set A we have that $\emptyset \subseteq A$ and $A \subseteq A$. ■

The term *cardinality* refers to the number of elements within a finite set A , denoted as $d(A)$. Accordingly, the cardinality of the empty set \emptyset is 0, as it contains no elements. For any two sets A and B , the notation $A \cup B$ signifies the *union* of A and B , whereas $A \cap B$ designates the *intersection* of A and B . When dealing with n sets, say A_1, A_2, \dots, A_n , their union and intersection are denoted as $\cup_{i=1}^n A_i$ and $\cap_{i=1}^n A_i$ respectively. For an arbitrary collection of sets indexed by I , we employ $\cup_{i \in I} A_i$ and $\cap_{i \in I} A_i$. In the context of an infinite collection of sets, the notations $\cup_i^\infty A_i$ and $\cap_i^\infty A_i$ are adopted.

On occasion, we will resort to the use of *Venn diagrams* as a visual means to represent sets, as exemplified in Figure 2.1.

Given the sets A and B , $A \setminus B$ is the *set difference*, and A^c is the *complement* set of A . The *De Morgan's laws* state that for every two sets A and B we have that $(A \cup B)^c = A^c \cap B^c$ and $(A \cap B)^c = A^c \cup B^c$.

Two sets A and B are *disjoint* if $A \cap B = \emptyset$. The sets A_1, A_2, \dots, A_n are disjoint if for every i and j such that $i \neq j$ we have that $A_i \cap A_j = \emptyset$. A *partition* of a set A is a collection of nonempty disjoint subsets of A_1, A_2, \dots, A_n of A such that $A = \cup_{i=1}^n A_i$. The *power set* $\mathcal{P}(A)$ is the set whose members are all possible subsets of A . If $d(A) = n$ then $d(\mathcal{P}(A)) = 2^n$.

Given any two sets A and B , we denote the *set difference* as $A \setminus B$, and the *complement* of the set A as A^c . *De Morgan's laws* articulate that for any pair of sets A and B , we have $(A \cup B)^c = A^c \cap B^c$ and $(A \cap B)^c = A^c \cup B^c$.

The term *disjoint* is applied to two sets A and B when their intersection $A \cap B = \emptyset$. We say that the sets A_1, A_2, \dots, A_n are disjoint if for every distinct pair i and j ($i \neq j$), we find $A_i \cap A_j = \emptyset$. A *partition* of a set A is an assembly of nonempty disjoint subsets A_1, A_2, \dots, A_n of A satisfying $A = \cup_{i=1}^n A_i$. The *power set* of A , denoted as $\mathcal{P}(A)$, is the collection of all conceivable subsets of A . If the cardinality of A is n , i.e., $d(A) = n$, then the cardinality of the power set of A is 2^n , thus expressed as $d(\mathcal{P}(A)) = 2^n$.

■ **Example 2.2** Given the set $A = \{1, 2, 3\}$, its power set is:

$$\mathcal{P}(A) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, A\}$$

■

Consider a non-empty set A and a collection \mathcal{F} of subsets of A . The pair (A, \mathcal{F}) is designated as an *field* over A if it fulfills the following conditions: it includes the empty set, denoted as $\emptyset \in \mathcal{F}$, it is closed under the operation of complementation, meaning for each $F \in \mathcal{F}$, the complement $F^c \in \mathcal{F}$, and it is closed under finite unions, which indicates $F_1 \cup \dots \cup F_n \in \mathcal{F}$ for all subsets $F_1, \dots, F_n \in \mathcal{F}$. Additionally, it can be demonstrated that a field also fulfills two further criteria: $A \in \mathcal{F}$, and it is closed under finite intersections, expressed as $F_1 \cap \dots \cap F_n \in \mathcal{F}$ for all subsets $F_1, \dots, F_n \in \mathcal{F}$.

Consider two elements, x and y . An *ordered pair*, symbolized as (x, y) , is a pairing of x and y in that order. By expanding this concept, an *n-tuple* — which can be visualized as an n -element ordered pair — is written as (x_1, \dots, x_n) . We define the *Cartesian product* of two sets A and B , which is symbolized as $A \times B$. This product is a set consisting of all possible ordered pairs (x, y) , where x belongs to set A and y belongs to set B . The Cartesian product can be extended to n sets, A_1, A_2, \dots, A_n , and is expressed as $A_1 \times A_2 \times \dots \times A_n$. Moreover, the *n-fold Cartesian product* of a set A with itself is represented as A^n .

Let R be a subset of the Cartesian product of set A with itself, i.e., $R \subseteq A \times A$. Such a subset is referred to as a *binary relation*, which can be represented as aRb , meaning that the ordered pair (a, b) is a member of R . A binary relation is deemed *reflexive* if for any element a in A , it holds that aRa . It is termed *symmetric* if for all a, b in A , the presence of aRb automatically implies bRa . The property of *antisymmetric* is attributed to a binary relation when the coexistence of aRb and bRa leads to the conclusion that $a = b$. It is considered *transitive* if for any a, b, c in A , the occurrence of aRb and bRc infers aRc . A binary relation is identified as *total* if for all a, b in A , either aRb or bRa holds. Binary relations can be extended to include two different sets A and B as a subset $R \subseteq A \times B$, and can also be adapted to *n-ary* relations represented as $R \subseteq A_1 \times A_2 \times \dots \times A_n$.

Let R be a binary relation that is a subset of the Cartesian product $A \times A$, i.e., $R \subseteq A \times A$. If this relation is reflexive, symmetric, and transitive, it is designated as an *equivalence relation*, typically symbolized by \sim . Within the context of an equivalence relation, two elements a, b from set A are deemed *equivalent* if the relation $a \sim b$ holds. The notion of an *equivalence class*, denoted as $[a]$, refers to the set of all elements in A that are equivalent to a particular element a . In other words, the equivalence class of a is defined as $[a] := \{b \in A : a \sim b\}$. An equivalence relation serves to partition the base set into what is known as the *quotient set*. Represented as A/\sim , the quotient set consists of all equivalence classes stemming from elements of A , that is, $A/\sim := \{[a] : a \in A\}$.

A binary relation that is reflexive, transitive, and antisymmetric is known as a *partial order*, often represented by the symbol \preceq . A set equipped with a partial order is referred to as a *partially*

ordered set, also abbreviated as *poset*. In the context of a poset, an element a from set A is considered *minimal* if there is no other element b in A for which $b \preceq a$. Similarly, an element a is labeled as *maximal* if no other element b in A exists such that $a \preceq b$. A relation that embodies reflexivity, transitivity, antisymmetry, and totality is designated as a *total order*, typically symbolized by \leq . A set that is coupled with a total order is identified as a *totally ordered set*. For any totally ordered set A , the *maximum* element is represented by $\max(A)$, meaning that $\max(A) \geq x$ holds for all x in A . Similarly, the *minimum*, represented by $\min(A)$, is defined as an element such that $\min(A) \leq x$ for all x in A .

■ **Example 2.3** Let R be a relation that is a subset of the Cartesian product of the set of natural numbers \mathbb{N} with itself, i.e., $R \subset \mathbb{N} \times \mathbb{N}$. In this relation, an ordered pair (a, b) belongs to R if a is a divisor of b . The set \mathbb{N} , when paired with relation R , forms a partially ordered set. In this context, the number 11 serves as a minimal element of R . This is because 11 is a prime number, which, by definition, can only be divided evenly by 1 and itself. ■

A *function* is defined as a binary relation $f \subseteq A \times B$, where for each element $x \in A$, there exists at most one $y \in B$ satisfying the condition $(x, y) \in f$. In this context, elements $(x, y) \in f$ are denoted by $f(x) = y$, with the function represented by $f : A \rightarrow B$. The set A is referred to as the *domain* of f , and B is the *codomain*. The set $\{y \in B : \exists x \in A, f(x) = y\}$ constitutes the *range* of f . If the relation is not defined for all $x \in A$, the function is termed *partial*, denoted by $f(x) \uparrow$ for undefined x in the function f .

■ **Example 2.4** In Section 4.4, we will discuss an alternate interpretation of a function as a procedure, or a series of steps, that assigns an element of B to each element of A . For instance, the subsequent C code delineates a partial function from \mathbb{R} to \mathbb{R} , characterized as partial due to $\text{inv}(0) \uparrow$:

```
double inv(double x) {
    return 1 / x;
}
```

A function is said to be *injective* if, for all elements x and y , $f(x) = f(y)$ implies $x = y$. A function is *surjective* when for every y , there exists at least one x such that $f(x) = y$. A function is described as *bijective* if it exhibits both injective and surjective properties. The *identity function* $I_A : A \rightarrow A$, determined by $f(a) = a$ for all $a \in A$, is bijective. These concepts of function, partial function, injective, surjective, and bijective can be extended to n -ary functions, represented as $f : A_1 \times A_2 \times \dots \times A_n \rightarrow B$.

The *inverse* function of a function f , represented by f^{-1} , is identified by $f(f^{-1}(x)) = f^{-1}(f(x)) = x$. Given two functions f and g , where the domain of f coincides with the range of g , we define the *composition* of f with g , represented by $f \circ g$, as $f \circ g = f(g(x))$.

An infinite set A is designated as *countable* if there is a bijective function that maps the elements of A onto the set of natural numbers \mathbb{N} . In contrast, a set is deemed *uncountable* if it is not finite and also not countable. We refer to a set as having *countably many* elements if it is either finite or countable.

■ **Example 2.5** Sets like \mathbb{N} , \mathbb{Z} and \mathbb{Q} are countable, while \mathbb{R} is not. ■

The *characteristic function* of a set A is denoted as $1_A : A \rightarrow \{1, 0\}$, where $1_A(x) = 1$ if $x \in A$ and 0 otherwise.

Considering a real number $x \in \mathbb{R}$, its *absolute value*, represented as $|x|$, is defined to be x if $x \geq 0$ and $-x$ if $x < 0$. The *ceil* function of x , denoted as $\lceil x \rceil$, is the smallest integer that is greater than or equal to x . The *floor* function of x , represented by $\lfloor x \rfloor$, is the largest integer that is less than

or equal to x . Given two positive integers a and b , the *modulo* operation of a and b , symbolized by $a \bmod b$, gives the remainder of the integer division of a by b .

For two functions f and g , defined as $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$, we assert that $f(n)$ is of *order of* $g(n)$, symbolized by $f(n) = O(g(n))$, if there exist positive integers c and m such that $f(n) \leq cg(n)$ for every integer $n \geq m$. If $f(n) = O(g(n))$, we denote g as an *upper bound* for f .

2.2 Strings and Languages

Consider a non-empty finite set $\mathcal{S} = \{s_1, s_2, \dots, s_q\}$, which we will refer to as the *alphabet*. The elements of this set are termed *symbols*. A *sequence* over \mathcal{S} is defined as an ordered arrangement of symbols $x_1x_2\dots x_n$ taken from \mathcal{S} . In the special case where the alphabet is the set $\mathcal{B} = \{0, 1\}$, such sequences are known as *binary sequences*. We use the term *string* to denote a finite sequence. This book predominantly focuses on binary strings.

The *length* of a string s , represented as $l(s)$, refers to the total number of symbols present in s . We use the notation λ to represent the *empty string*, which is defined as the unique string over \mathcal{S} that has a length of 0. Given a symbol $x \in \mathcal{S}$, the string comprising x repeated n times is denoted by x^n . If $s = x_1x_2\dots x_n$ constitutes a string, its *reverse*, designated as s^R , is $x_nx_{n-1}\dots x_1$.

The set of all strings $s_1s_2\dots s_n$ of length n over the alphabet \mathcal{S} is denoted by \mathcal{S}^n ¹. We denote by \mathcal{S}^+ the union of all \mathcal{S}^n for $n \geq 1$, and by \mathcal{S}^* the set $\mathcal{S}^+ \cup \{\lambda\}$. Note that all strings in \mathcal{S}^* have finite lengths. The term *Kleene closure* refers to \mathcal{S}^* .

■ **Example 2.6** The following relations hold: the cardinality of the set of binary strings of length n is $d(\{s \in \mathcal{B}^* : l(s) = n\}) = 2^n$, and the cardinality of the set of binary strings of length up to n is $d(\{s \in \mathcal{B}^* : l(s) \leq n\}) = 2^{n+1} - 1$. ■

Given two strings s and t from \mathcal{S}^* , the *concatenation* of s and t , symbolized as st , is the sequence that results from placing the sequence of symbols in t immediately after the sequence of symbols in s . Consequently, the length of the concatenated string, $l(st)$, is the sum of the lengths of s and t . This suggests that \mathcal{S}^* is closed under the concatenation operation. Furthermore, the set \mathcal{S}^* , together with the operation of concatenation, forms a *free monoid*. This implies that concatenation is associative $(st)r = (st)r$, and that there is an identity element, specifically, the empty string λ , for which $\lambda a = a\lambda = a$ holds.

A string s is termed a *substring* of t if there exist strings u and v (which may be empty) such that $t = usv$. If there exists a string u such that $t = su$, s is considered a *prefix* of t , which is notated as $s <_p t$. A subset $S \subset \mathcal{S}^*$ is described as *prefix-free* if, for any $s, t \in S$, $s = t$ whenever $s <_p t$. Let $S, T \subset \mathcal{S}^*$ be two sets of strings, the (left) *quotient* $S^{-1}T$ is defined as the residual words obtained from T by removing some prefix in S ; formally, $S^{-1}T = \{t \mid st \in T \wedge s \in S\}$.

We denote the *self-delimited* form of a string $s \in \mathcal{S}^*$ by \bar{s} , and define it as $\bar{s} = 1^{l(s)}0s$. Consequently, the length of \bar{s} , $l(\bar{s})$, is twice the length of s plus one, i.e., $l(\bar{s}) = 2l(s) + 1$.

■ **Example 2.7** The set $\bar{\mathcal{S}}^*$, which comprises all the self-delimited strings from \mathcal{S}^* , is prefix-free. ■

In cases where \mathcal{S} is a totally ordered set, we are able to define a total order on \mathcal{S}^* . This ordering, termed *shortlex ordering*, arranges sequences primarily by length, positioning the shortest sequences first. Sequences of identical length are further sorted according to lexicographical order.

■ **Example 2.8** Given $S = \{a, b, c\}$ with $a < b < c$, the shortlex order on \mathcal{S}^* generates the relations $\lambda < a < b < c < aa < ab < \dots < cc < aaa < aab < \dots < ccc < \dots$ ■

For any arbitrary object O , we utilize the notation $\langle O \rangle$ to represent its string representation, presupposing the existence of a standard encoding method. For objects O_1, O_2, \dots, O_k , the

¹It is important to avoid confusing the set of strings of length n over an alphabet \mathcal{S}^n with the n -fold Cartesian product of a set \mathcal{S}^n . The use of calligraphic fonts helps distinguish between alphabets and other sets.

notation $\langle O_1 O_2 \dots O_k \rangle$ denotes the concatenation of the string representations of these objects: $\langle O_1 \rangle \langle O_2 \rangle \dots \langle O_k \rangle$. We employ the notation $\langle O_1, O_2, \dots, O_k \rangle$ to indicate the concatenation of the representations of these objects in a manner that allows for decoding and unique identification of all objects. As an example, $\langle O_1, O_2, \dots, O_k \rangle$ might be represented as $\langle \bar{O}_1 \rangle \langle \bar{O}_2 \rangle \dots \langle \bar{O}_k \rangle$.

■ **Example 2.9** Natural numbers can be represented by binary strings via the following encoding method: $\langle 0 \rangle = \lambda$, $\langle 1 \rangle \rightarrow 0$, $\langle 2 \rangle \rightarrow 1$, $\langle 3 \rangle \rightarrow 00$, $\langle 4 \rangle \rightarrow 01$, $\langle 5 \rangle \rightarrow 10$, $\langle 6 \rangle \rightarrow 11$, $7 \rightarrow 000$, and so on. Therefore, the pair of numbers $\langle 3, 7 \rangle$ would be represented as 110001110000 . Given this particular encoding, it follows that $l(\langle n \rangle) = \lfloor \log_2(n+1) \rfloor$. ■

A *language* L over an alphabet \mathcal{S} is a set of strings $L \subset \mathcal{S}^*$. The elements of L are called *words*. The *empty language* is the language that contains no words $L = \emptyset$.

A *language*, denoted by L , over an alphabet \mathcal{S} , is defined as a subset of strings such that $L \subseteq \mathcal{S}^*$. The individual elements of L are termed as *words*. The unique language that does not contain any words is referred to as the *empty language*, and is expressed as $L = \emptyset$.

Consider two languages L_1 and L_2 over a common alphabet \mathcal{S} . There are several standard operations that can be applied to these languages, including: language union $L_1 \cup L_2 = \{w \in \mathcal{S}^* \mid w \in L_1 \text{ or } w \in L_2\}$, language intersection $L_1 \cap L_2 = \{w \in \mathcal{S}^* \mid w \in L_1 \text{ and } w \in L_2\}$, language complement $\overline{L_1} = \{w \in \mathcal{S}^* \mid w \notin L_1\}$, and Kleene closure $L_1^* = \{\lambda\} \cup \{wz \mid w \in L_1 \text{ and } z \in L_1^*\}$.

Languages can be systematically generated using a finite set of string rewriting rules known as grammars. A *grammar*, denoted as G , is a 4-tuple (N, Σ, P, S) , where: $N \subseteq \mathcal{S}$ is a finite set of *nonterminal symbols*; $\Sigma \subseteq \mathcal{S}$ is a finite set of *terminal symbols*; P is a finite set of *production rules* in the form $(\Sigma \cup N)^* N (\Sigma \cup N)^* \rightarrow (\Sigma \cup N)^*$; and $S \in N$ is a special *start symbol*. Each production rule transforms one string of symbols into another, starting with the designated start symbol.

■ **Example 2.10** Let us consider the alphabet $\mathcal{S} = \{S, a, b\}$, and define the grammar (N, Σ, P, S) where $N = \{S\}$, $\Sigma = \{a, b\}$, $P = \{S \rightarrow aSb, S \rightarrow ba\}$, and the start symbol is $S \in N$. This grammar generates the language $L = \{a^n bab^n \mid n \geq 0\} = \{ba, abab, aababb, aaababbb, \dots\}$. ■

The *Chomsky hierarchy* serves as a categorization system for grammars, based on their expressive capacity or the range of languages they can generate. The hierarchy, listed from the most to the least restrictive grammars, is defined as follows (where a represents a terminal symbol, A, B are non-terminal symbols, and α, β , and γ denote strings composed of either terminal or non-terminal symbols):

Type-3 Also known as *regular grammars*. For these grammars, the left-hand side of every production rule consists solely of a single nonterminal symbol. The right-hand side may either be an empty string, a single terminal symbol, or a single terminal symbol followed by a nonterminal symbol. Formally, the rules are $(A \rightarrow \lambda, A \rightarrow a, A \rightarrow aB)$.

Type-2 These are referred to as *context-free grammars*. Here, the left-hand side of each production rule is comprised of only a single nonterminal symbol, symbolically expressed as $(A \rightarrow \alpha)$.

Type-1 These are *context-sensitive grammars*. In this case, the left and right sides of the production rules may be embedded within a context of terminal and nonterminal symbols. This can be written as $(\alpha A \beta \rightarrow \alpha \gamma \beta)$.

Type-0 The *recursively enumerable grammars* fall in this category, and they do not impose any constraints on the production rules ($\gamma \rightarrow \alpha$).

■ **Example 2.11** The grammar discussed in Example 2.10 is classified as a Type-2, or context-free grammar. ■

The *Backus–Naur form* is a notation system specifically designed for context-free grammars. This notation is frequently employed in the field of computer science to formally outline the syntax of programming languages and communication protocols. A Backus–Naur form consists of a set of production rules structured in the following way:

```
<symbol> ::= __expression__
```

Here, `<symbol>` stands for a non-terminal symbol, `__expression__` refers to a string of terminal or non-terminal symbols, and `::=` indicates that the symbol on the left-hand side should be substituted with the expression on the right. Several `__expression__`s can be integrated into a single production rule by separating them with a vertical bar `|`, suggesting that any one of them can be selected for substitution. Symbols that are not found on a left-hand side of a production rule are considered terminal symbols. Conversely, those that appear on a left-hand side are non-terminal symbols and are invariably enclosed within the pair `<>`. The non-terminal symbol of the first production rule is identified as the start symbol.

- **Example 2.12** The grammar discussed in Example 2.10 can be reformulated in Backus-Naur form using the subsequent production rule:

```
<string> ::= a <string> b | ba
```

■

2.3 Counting Methods

Combinatorics, a specialized branch of mathematics, primarily focuses on the investigation of discrete objects and their mutual relationships. The core aspects of combinatorics include the counting, arranging, and selection of these objects, supplemented by the methodologies employed for achieving these objectives. It presents an array of robust tools essential for handling extensive collections of objects exhibiting certain characteristics. In this section, we shall focus on revisiting the significant outcomes of combinatorics from the perspective of sets and ordered lists.

The *multiplication rule* is a fundamental theorem that specifies the number of potential outcomes in the Cartesian product of sets. According to this rule, if there are k sets A_1, A_2, \dots, A_k , and each set contains n_i elements ($i = 1, \dots, k$), then the Cartesian product $A_1 \times A_2 \times \dots \times A_k$ encapsulates a total of $n_1 n_2 \dots n_k$ elements. Particularly, if a set A consists of n elements, then A^k includes n^k elements.

The *inclusion-exclusion principle* provides the count of the union of several sets, given the individual size of each set, and the size of every feasible intersection of these sets. Given k sets A_1, A_2, \dots, A_k , the expression is as follows:

$$\begin{aligned} d\left(\bigcup_{i=1}^k A_i\right) &= \sum_{i=1}^k d(A_i) - \sum_{i < j} d(A_i \cap A_j) + \sum_{i < j < l} d(A_i \cap A_j \cap A_l) - \\ &\quad - \sum_{i < j < l < m} d(A_i \cap A_j \cap A_l \cap A_m) + \dots + (-1)^{k+1} d(A_1 \cap A_2 \cap \dots \cap A_k) \end{aligned}$$

Permutations denote the number of ways in which a set's elements can be arranged. Suppose A is a set with n elements, the number of permutations of n elements taken k at a time, represented as $P_{n,k}$, is $P_{n,k} = n(n-1)\dots(n-k+1)$. When $k = n$, the permutations are calculated as $P_{n,n} = n(n-1)\dots 1 = n!$, where $n!$ is known as n factorial.

The *pigeonhole principle* is a simple yet powerful concept that states if you have more pigeons than pigeonholes, then there must be at least one pigeonhole with more than one pigeon. In a more mathematical sense, if you have n items to place into m containers and $n > m$, then at least one container must contain more than one item.

For performing vast computations, an approximation of $n!$, commonly known as *Stirling's formula*, proves highly effective:

$$\log(n!) \approx \frac{1}{2} \log(2\pi) + \left(n + \frac{1}{2}\right) \log(n) - n$$

■ **Example 2.13** Consider the set $\{a, b, c\}$. There are six unique permutations of its elements: $[a, b, c], [a, c, b], [b, a, c], [b, c, a], [c, a, b], [c, b, a]$. Each permutation signifies a distinct order of the elements in the original set. ■

Numerous counting problems revolve around determining the quantity of subsets of a certain size present within a given set. Given a set with n elements, the total possible subsets amount to 2^n , inclusive of the empty set and the set itself. The quantity of subsets of size k , also known as the number of *combinations* of k elements from a set of n , denoted by $C_{n,k}$, is computed using the formula $C_{n,k} = \frac{P_{n,k}}{k!} = \frac{n!}{k!(n-k)!}$. The symbol $\binom{n}{k}$ also represents the number $C_{n,k}$, which is known as the *binomial coefficient*. We know that for all n , $\binom{n}{0} = \binom{n}{n} = 1$, and for all n and $k = 0, 1, \dots, n$, $\binom{n}{k} = \binom{n}{n-k}$. Moreover, $\binom{n}{k} = 0$ for $k > n$.

■ **Example 2.14** Take for instance the set $\{a, b, c, d\}$. There exist 4 combinations of size 3: $[a, b, c]$, $[a, b, d]$, $[a, c, d]$, and $[b, c, d]$. Combinations do not consider order, hence $[a, c, d]$ and $[d, c, a]$ are deemed the same combination. ■

The *multinomial coefficient*, an extension of the binomial coefficient to more than two categories or types, signifies the number of ways a set of objects can be partitioned into a fixed number of subsets, each of which contains a specific number of objects. Assuming we have a set with n elements, which can be partitioned into k subsets of sizes n_1, n_2, \dots, n_k , respectively. The multinomial coefficient, symbolized as $\binom{n}{n_1, n_2, \dots, n_k}$, represents the number of feasible partitions, and is computed as $\binom{n}{n_1, n_2, \dots, n_k} = \frac{n!}{n_1!n_2!\dots n_k!}$.

2.4 Matrices

A *matrix*, denoted by A , of order $m \times n$ is composed of a sequence of mn scalars. These scalars are arranged in a rectangular array consisting of m *rows* and n *columns*, as depicted below:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

The *entry* a_{ij} denotes the element of A found at the i -th row and j -th column. The *set of all matrices* of order $m \times n$ is symbolized as $\mathcal{M}_{m \times n}$. A *row matrix* belongs to the set $\mathcal{M}_{1 \times n}$, whereas a *column matrix* is a member of $\mathcal{M}_{m \times 1}$. A *square matrix* is any matrix within the set $\mathcal{M}_{n \times n}$. The entries a_{ii} compose the *main diagonal* of a square matrix. A *diagonal matrix* is distinguished by having all of its entries outside the main diagonal as zero. The *identity matrix*, denoted by I , is a diagonal square matrix with all entries on the main diagonal equal to 1.

The *transpose* of a matrix $A \in \mathcal{M}_{m \times n}$ is defined as the matrix $A^T \in \mathcal{M}_{n \times m}$, with entries at position (i, j) mirroring the entries at position (j, i) in A . If $A = A^T$, then the matrix A is classified as a *symmetric matrix*. A *submatrix* of a matrix is obtained by eliminating any selection of rows and/or columns.

■ **Example 2.15** Take for instance the square matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$. Here, the entry located at position $(2, 3)$ has the value 6, the main diagonal is made up of the numbers 1, 5, and 9. The transpose of the matrix is $A^T = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$, and the matrix $B = \begin{pmatrix} 1 & 3 \\ 4 & 6 \end{pmatrix}$ is identified as a submatrix of A . ■

The addition of two matrices A and B of identical size results in a new matrix $A + B$. Each entry (i, j) of this matrix is given by $(A + B)_{ij} = a_{ij} + b_{ij}$. The operation of matrix addition exhibits

associativity, i.e., $(A + B) + C = A + (B + C)$, and commutativity, i.e., $A + B = B + A$. It has a neutral element, such that $A + 0_{m \times n} = A$, and an inverse element, $A + (-A) = 0_{m \times n}$.

The product of a scalar λ and a matrix A yields another matrix, denoted λA , where each entry (i, j) is $(\lambda A)_{ij} = \lambda a_{ij}$. This scalar multiplication operation is distributive relative to the addition of matrices, as $(\alpha(A + B)) = \alpha A + \alpha B$, and to the addition of scalars, as $(\alpha + \beta)A = \alpha A + \beta B$. It is also associative with scalar multiplication, such that $(\alpha\beta)A = \alpha(\beta A)$, and has a unit element, as $1A = A$.

The product of two matrices $A_{m \times n}$ and $B_{n \times p}$ results in a matrix $AB_{m \times p}$, where each entry (i, j) is given by $(AB)_{ij} = \sum_{k=1}^n a_{ik}b_{kj}$. The operation of matrix multiplication is associative, as $(AB)D = A(BD)$, and it possesses a left neutral element $AI_n = A$ and a right neutral element $I_m A = A$. It is associative with respect to scalar multiplication, such that $\alpha(AB) = (\alpha A)B = A(\alpha B)$, and is distributive with respect to matrix addition, both from the right $A(B + C) = AB + AC$ and from the left $(B + C)D = BD + CD$.

Additionally, the transpose operation satisfies the following properties: $(A + B)^T = A^T + B^T$, $(\lambda A)^T = \lambda A^T$, and $(AB)^T = B^T A^T$.

■ **Example 2.16** Given the matrices $A = \begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix}$ and $B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$ we have that $A + B = \begin{pmatrix} 6 & 8 \\ 10 & 13 \end{pmatrix}$, $2A = \begin{pmatrix} 2 & 4 \\ 6 & 10 \end{pmatrix}$ and $AB = \begin{pmatrix} 19 & 22 \\ 50 & 58 \end{pmatrix}$. ■

A square matrix A is *invertible* or *non-singular* if there exists a matrix B such that $AB = BA = I$. If A is non-singular, B is unique and is called the *inverse matrix* of A , denoted by A^{-1} . A matrix A is *orthogonal* if its transpose is equal to its inverse $A^T = A^{-1}$; the columns and rows of a orthogonal matrix are called *orthonormal vectors*.

The *determinant* is a special scalar value that can be computed from the elements of a square matrix. The determinant is denoted as $\det(A)$ for a matrix A . The determinant of a square matrix can be computed using the Leibniz formula, which is given as:

$$\det(A) = \sum_{\sigma \in S_n} \text{sgn}(\sigma) \cdot a_{1,\sigma(1)} \cdot a_{2,\sigma(2)} \cdots a_{n,\sigma(n)}$$

where S_n denotes the set of all permutations of the numbers 1 to n , and $\text{sgn}(\sigma)$ is the signature of the permutation σ , being +1 for even permutations and -1 for odd permutations. The determinant is nonzero if and only if the matrix is invertible

■ **Example 2.17** The computation of the determinant of a 3x3 matrix $A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$ is given by:

$$\det(A) = aei + bfg + cdh - ceg - bdi - afh$$

■

For a given matrix A , the *rank*, denoted as $\text{rank}(A)$, is the maximum number of linearly independent rows or columns within the matrix.

A number λ , and a non-zero vector \mathbf{v} satisfying $A\mathbf{v} = \lambda\mathbf{v}$ are called an *eigenvalue* and an *eigenvector* of A , respectively.

Matrix decomposition is the process of rendering a matrix into a more easily accessible form, meanwhile certain properties, like the determinant or the rank, are preserved. The *singular value decomposition* of a matrix A of order $m \times n$ is a factorization of the form $A = U\Sigma V^T$ where U is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ non-negative diagonal matrix of rectangular shape, and V^T is the transpose of an $n \times n$ orthogonal matrix.



Figure 2.2: An Example of Graph

2.5 Graphs

A *graph*² G is represented as an ordered pair (V, E) , comprising a set V , referred to as *vertices*, and a set E , denoted as *edges*. The members of E take the form of unordered pairs $\{u, v\}$, constituting distinct vertices $u, v \in V$ (loops are not permitted). Vertices u and v are described as *adjacent* when an edge $\{u, v\} \in E$ exists, consequently, they are termed the *endpoints* of that edge. If the set V is infinite, the graph is classified as an *infinite graph*. This book, however, focuses exclusively on finite graphs. Given that $G = (V, E)$ is a graph, its *adjacency matrix* is a square $d(V) \times d(V)$ matrix A whereby element A_{uv} equals 1 if $\{u, v\} \in E$ and 0 otherwise.

Graphs are typically illustrated as a collection of dots representing vertices, linked by lines denoting edges.

■ **Example 2.18** If $V = \{a, b, c, d\}$ and $E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}\}$, the graph $G = (V, E)$ is represented in Figure 2.2. ■

When a vertex v serves as an endpoint of an edge e , it is said that e is *incident* upon v . The *degree* of a vertex v , denoted as $\deg(v)$, corresponds to the count of edges incident on v . A vertex with degree zero is labeled as *isolated*, while a vertex with degree one is termed *pendant*. The *neighborhood* of a vertex v , signified by $N(v)$, includes all vertices that are adjacent to v . If $A \subset V$, the neighborhood of A is given by $N(A) = \cup_{v \in A} N(v)$. A *path* in a graph consists of a series of unique vertices $\{v_0, v_1, \dots, v_k\}$ such that v_i and v_{i+1} are adjacent for each $1 \leq i < k$. A path is known as a *simple path* if no vertex is repeated. A graph is considered *connected* when a path exists between any two vertices. If $v_0 = v_k$, the path is defined as a *cycle*. A cycle is called a *simple cycle* if it consists of at least three vertices and only the first and last vertices are repeated.

■ **Example 2.19** In a given graph $G = (V, E)$, the *handshaking theorem* posits that $\sum_{v \in V} \deg(v) = 2m$, where $m = d(E)$, given that each edge has two endpoints. ■

If the vertex pairs u, v are arranged in ordered pairs, the graph is termed a *directed graph*. In this case, u is referred to as the *initial vertex*, while v is known as the *terminal vertex*. Given that G is a directed graph, the *in-degree* of a vertex v , signified by *indeg*(v), corresponds to the count of edges in which v serves as a terminal vertex. The *out-degree*, denoted by *outdeg*(v), refers to the number of edges where v is the initial vertex. A directed graph is *strongly connected* if a directed path connects each pair of vertices. Directed graphs are typically illustrated with arrows, rather than lines, to represent edges.

A graph G is classified as *bipartite* if the vertex set V can be partitioned into two subsets V_1 and V_2 such that every edge of G links a vertex from V_1 to one from V_2 . Bipartite graphs are usually denoted as $G = (V_1, V_2, E)$. The degree of vertices in a bipartite graph adheres to the *degree sum formula*, $\sum_{u \in V_1} \deg(u) = \sum_{v \in V_2} \deg(v) = d(E)$.

A graph $G(V', E')$ is a *subgraph* of $G(V, E)$ if V' is a subset of V and E' is a subset of E with

²The definition of a graph stated here corresponds to the definition of a *simple graph* as found in typical discrete mathematics literature.



Figure 2.3: An Example of a Tree

endpoints that belong to V' . A graph G is termed a *labeled graph* if its edges and/or vertices are assigned specific data. Specifically, if each edge e of G is allocated a nonnegative number $w(e)$, then $w(e)$ is referred to as the *weight* of e .

A specific type of graph that plays a fundamental role in this book is the *tree*. Defined as a non-empty graph, a tree ensures any pair of vertices are interconnected by a singular, unique path. A tree always includes a specially designated vertex termed the *root*, and every edge of the tree is oriented away from the root.

■ **Example 2.20** An alternative definition of trees, based on set theory, posits that a tree is a partially ordered set $(T, <)$ such that for each $t \in T$, the set $S = \{s \in T : s < t\}$ possesses a least element – an element smaller than all other elements in S . ■

Given a tree T , for any vertex v other than the root, the *parent* of v is the single, unique vertex u such that an edge directly links u to v . Conversely, if u is the parent of v , v is then identified as a *child* of u . Any other vertex within the tree sharing the same parent as v is considered a *sibling* of v . The *ancestors* of a vertex encompass all vertices along the path from the root to the given vertex, excluding the vertex itself but including the root. The *descendants* of a vertex v include those vertices that count v as an ancestor. A vertex with no children is labeled as a *leaf*, whereas vertices possessing children are referred to as *branches*. The *depth* of a vertex v is determined by the length of the unique path leading from the root to v . The *height* of a tree is the maximum among the depths of all its vertices.

■ **Example 2.21** For the tree illustrated in Figure 2.3, the root vertex is a ; c serves as a parent of d , making d a child of c ; d and g are siblings; the ancestors of d include a and c ; the descendants of c comprise d , e , and f ; leaf nodes in the tree are b , e , f , and g ; a and c are branches; the depth of d is 3; the height of the tree is 4. ■

Given a vertex v in a tree, the *subtree* with v as its root is the subgraph within the tree that includes v , its descendants, and all edges connected to these descendants. A tree is termed a *k-ary tree* if each branch houses no more than k children. If every branch comprises exactly k children, the tree is then labeled a *full k-ary tree*. A *k-ary tree* where $k = 2$ is specifically referred to as a *binary tree*. A *k-ary tree* of height h is deemed *balanced* if all its leaves are located at a depth of h or $h - 1$.

■ **Example 2.22** A tree composed of n vertices includes $n - 1$ edges. A full *k-ary tree* featuring i

branches hosts $m = ki + 1$ vertices. ■

The procedure of visiting each node in a tree exactly once is defined as *tree traversal*. The classifications of tree traversals are determined by the order of node visits, namely, *depth-first* and *breadth-first* order. In a depth-first traversal, the algorithm initiates at the root node and ventures as far as possible along each branch before transitioning to the next sibling. There are three standard strategies for processing nodes within the tree: *in-order*, *pre-order*, and *post-order*.

The following snippet of code, resembling C language syntax, demonstrates the usage of a recursive pre-order depth-first traversal algorithm to print a binary tree:

```
void print_tree(binary_tree *tree) {
    if (!is_empty(tree)) {
        printf("%c\n", tree->node); /* print node */
        print_tree(tree->left_branch); /* process left branch */
        print_tree(tree->right_branch); /* process right branch */
    }
}
```

Conversely, in a breadth-first traversal, the algorithm commences at the tree root and explores all nodes at the current depth before progressing to nodes at the subsequent depth level. Implementing depth-first tree traversal algorithms necessitates the employment of sophisticated data structures. For an example of such algorithms, please consult the references section.

■ **Example 2.23** In the case of the tree delineated in Example 2.18, a pre-order depth-first traversal would yield the string "abcdefg". Conversely, a pre-order breadth-first traversal would generate the string "abcdgef". ■

References

The book "Discrete Mathematics" by Johnsonbaugh [[johnsonbaugh2009discrete](#)] is tailored for undergraduate students taking a one- or two-semester course in discrete mathematics, thoroughly covering key topics in the field. "Introduction to the Theory of Computation" by Sipser [[sipser2012introduction](#)] offers a comprehensive, clear, and student-centric introduction to the latest computational theory topics and methodologies. It is highly acclaimed for its in-depth exploration of automata, formal languages, and complexity theory. "Introduction to Algorithms" by Cormen et al. [[cormen1990introduction](#)], commonly known by the acronym "CLRS" derived from the authors' initials, is a seminal work in the algorithms field, encompassing a broad spectrum of topics, including graph algorithms. This book is both extensive and rigorous. Finally, "Matrix Computations" by Golub and Van Loan [[golub2013matrix](#)] delves into various topics related to matrices, with an emphasis on matrix computations - an area of particular interest to those engaged in computational studies.

3. Discrete Probability

*Mathematics may be defined as the subject in which
we never know what we are talking about,
nor whether what we are saying is true.*

Bertrand Russell

TODO: Change image and quote.

Probability theory is the branch of mathematics that studies random experiments and random phenomena. Probability assigns a numerical description to all the possible outcomes of an experiment, according to how likely is that these outcomes will occur. Even if the outcome of an experiment cannot be determined in advance, we can study its properties with probability theory and come to relevant results and conclusions. For example, we cannot predict the next number that will appear in a lottery game, but probability theory can help us to understand why it is not a good strategy to spend all our savings on lottery tickets with the goal of becoming rich.

Probability theory provides the mathematical foundations for statistical inference, one of the most relevant approaches we have today for gathering knowledge based on the analysis of the results of carefully designed experiments. Probability theory also provides the foundation of machine learning, that is, the analysis of large volumes of data to derive intelligent algorithms.

In this chapter we are going to focus in the area of discrete probability. In this version of the theory, the possible outcomes of an event are finite, or at most, countably infinite. We are interested in the area of discrete probability, first because its applications in practice to the area of learning from data, and second, because discrete probability has some very interesting connections to theories used in this book: the length of optimal codes, the probability that a random machine will halt, and the derivation of universal distributions based on Kolmogorov complexity. All of these connections are relevant in our theory of nescience. We are going to study probability theory from a formal, axiomatic, point of view. We will start by formulating a very basic collection of fundamental axioms, and then we will derive the major results and properties from them.

This chapter is a very brief overview of probability theory. We will cover only the most important techniques and results. The contents have been selected based on their applicability

to the theory of nescience. For example, moment generating functions are not covered. For a more comprehensive introduction to probability theory, see the References section at the end of the Chapter.

3.1 Foundations

The concept of *probability* represents a profound intellectual challenge. Let us consider an instance where a dice is rolled and the objective is to compute the probability of an even number being the outcome. The dice comprises six distinct outcomes, and given that half of these are even numbers, we posit that the probability of yielding an even number is $3/6$ or equivalently $1/2$. This embodies the *classical interpretation* of probability which asserts that in an experiment where all finite potential outcomes possess an equal likelihood of occurrence, the probability of an event equates to the count of favorable instances over the total number of instances. This interpretation, however, confronts the issue of circularity in its definition as "equally likely" essentially amounts to "possessing the same probability". An alternative method for probability assignment might be the implementation of the *principle of indifference*, which postulates that in the absence of any relevant evidence, all potential outcomes should possess identical probability. This principle, however, encounters a predicament when there exists evidence that contradicts the presumption of equivalence amongst all outcomes. For instance, how should probability be assigned when knowledge of a loaded dice suggests that not all sides possess an equal likelihood of occurrence?

The *frequentist interpretation* of probability posits that one should roll the dice multiple times and contrast the frequency of even numbers with the total number of rolls. The fundamental notion is to execute the experiments repeatedly under similar conditions and assign the relative frequency of each outcome as its probability. This interpretation, however, faces two primary limitations. Firstly, the definition of repeating an experiment under "similar conditions" remains vague; if the conditions were truly identical, the results across all trials would invariably be the same. Secondly, the notion of a "large number of times" is undefined (technically, the experiment should be executed an infinite number of times). From a practical standpoint, implementing the frequentist interpretation is fraught with challenges. Some experiments, such as predicting the probability of a candidate winning an election, cannot be repeated numerous times. Furthermore, probability is defined in the context of a sequence of experiments, hence precluding the computation of the probability of a solitary outcome. Lastly, this interpretation necessitates the existence of a relative frequency limit, a condition which is not always satisfied, as illustrated by financial time series.

The *subjective interpretation*, representing a third approach to the concept of probability, suggests assigning probabilities to each event that reflect our degree of belief: the higher our conviction in the event's occurrence, the greater its assigned probability. However, it's important to note that not all potential probability allocations are viable; certain coherence rules need to be satisfied. For instance, when placing bets on the outcome of a dice roll, an assignment of probabilities that ensures a complete loss of money - a scenario known as a *dutch book* - would contravene the conditions of the subjective interpretation. It transpires that the conditions both necessary and sufficient to ensure a fair bet align with the axioms of probability introduced subsequently. Hence, we are free to assign any probabilities we desire to events, provided they remain consistent with the axioms of probability. A key drawback of the subjective interpretation is the inherent variability in individuals' degrees of belief. The *Bayesian interpretation* of probability offers a solution: we commence with a provisional assignment of probabilities and, upon accruing further evidence, adjust our degree of belief or probability accordingly. With the accumulation of more evidence, estimated probabilities will converge to the true probabilities. Regardless, the task of assigning probabilities to an infinite number of events is typically unattainable for humans in general.

Currently, the notion of probability is defined axiomatically via the *axiomatic interpretation*. This implies that we abandon attempts to explicitly define probability and instead accept some of its properties as inherently true. Intuitively, a probability should be a value between 0 and 1, wherein an event with a zero probability is deemed impossible, and an event with a probability of one is certain to occur. Additional properties are required of probabilities. For instance, should two events A and B with probabilities $P(A)$ and $P(B)$ respectively, be disjoint, the probability of either A or B occurring should be $P(A) + P(B)$. If A and B could occur simultaneously and are independent (however that is defined), the probability of both events occurring concurrently should be $P(A)P(B)$. Moreover, the probability of A occurring given that B has already happened should be the fraction of the probability of A that intersects with B . Thus, anything that satisfies these properties could be considered a probability. The issue with the axiomatic interpretation is that an abundance of constructs meet these requirements, including physical quantities such as normalized mass or normalized volume.

Probability theory is fundamentally concerned with the task of assigning a numerical value to specific events drawn from a sample space. The term "event" in this context may be somewhat misleading, as it intimates the occurrence of something, which is not always applicable. For instance, consider the sample space of all possible outcomes when tossing a fair coin. A subset of this sample space could be the empty set, which represents no coin toss happening at all. In the conventional understanding of an "event", this scenario is rather counterintuitive, as it does not correspond to something "happening". Nonetheless, for the sake of clarity, we will continue to employ the term "events" to designate what essentially are subsets.

Definition 3.1.1 Given (Ω, \mathcal{A}) as a field over a non-empty discrete set, Ω is referred to as the *sample space*, its constituents are termed *outcomes*, and the components of \mathcal{A} are referred to as *events*. Specifically, Ω is designated the *certain event*, while the empty set \emptyset is deemed the *impossible event*.

As previously discussed in Section 2.1, given that (Ω, \mathcal{A}) is a field, we can deduce that $\Omega \in \mathcal{A}$ and that $\emptyset \in \mathcal{A}$. Additionally, the union of a finite collection of events constitutes an event $A_1 \cup A_2 \cup \dots \cup A_n \in \mathcal{A}$, and the intersection of a finite collection of events is likewise deemed an event $A_1 \cap A_2 \cap \dots \cap A_n \in \mathcal{A}$.

As alluded to in the introduction of this chapter, our principal interest lies in discrete mathematics, and hence, we will largely focus on probabilities pertaining to discrete sets (be they finite or countably infinite). An extension of the concept of probability to continuous sets would necessitate the utilization of σ -algebras of sets instead of fields and the application of measure theory. For example, consider a sample space representing the possible outcomes of a roll of a continuous, rather than discrete, die. Instead of the discrete outcomes 1, 2, 3, 4, 5, and 6, we might have any real number between 1 and 6. In this case, it doesn't make sense to talk about the probability of the outcome being exactly 2.5 (or any other specific real number), as there are infinitely many possible outcomes. Instead, we would talk about the probability of the outcome being in a certain interval, such as between 2 and 3.

The prevailing axiomatization utilized in the realm of probability theory is encapsulated within the framework of the *Kolmogorov axioms*.

Definition 3.1.2 (Kolmogorov's Axioms) A *probability* is a real number $P(A) \in \mathbb{R}$ allocated to each event $A \in \mathcal{A}$ in the field (Ω, \mathcal{A}) . This allocation adheres to the following axioms:

Axiom 1 Each probability is nonnegative: $P(A) \geq 0$.

Axiom 2 The probability of the certain event is one: $P(\Omega) = 1$.

Axiom 3 For any finite sequence of disjoint events A_1, A_2, \dots, A_n , the probability of the union of these events is the sum of their probabilities: $P(\bigcup_{i=1}^n A_i) = \sum_{i=1}^n P(A_i)$.

The triplet (Ω, \mathcal{A}, P) constitutes what is known as a *probability space*.

Despite their significance, the Kolmogorov axioms encounter certain complexities. One challenge is that they are incompatible with the reduction to first-order logic, as elaborated in Appendix B. This incompatibility arises due to the fact that real numbers, to which probabilities are assigned, are not describable within the ambit of first-order logic. Furthermore, the axioms do not provide explicit guidance on the assignment of probabilities to events. Essentially, they set out the foundational principles that probabilities must conform to but do not dictate the process of how these probabilities should be determined.

■ **Example 3.1.1** Consider a sample space Ω composed of n equally probable elements. If we have an event $A \subset \Omega$ comprised of $d(A) = m$ elements, then the probability of event A can be represented as $P(A) = m/n$. ■

We now venture to establish certain fundamental theorems concerning probabilities, beginning with the calculation of the complement of an event, which represents the probability of an event not occurring.

Proposition 3.1.1 For each event A , it holds true that $P(A^c) = 1 - P(A)$.

Proof. Sets A and A^c are disjoint, and their union $A \cup A^c$ equals Ω . By applying Axiom 3, we infer that $P(A \cup A^c) = P(A) + P(A^c)$, and by applying Axiom 2, we conclude that $P(A \cup A^c) = P(\Omega) = 1$. Therefore, we can assert that $P(A) + P(A^c) = 1$. ■

As a direct consequence of the aforementioned proposition, we can deduce the probability of the impossible event.

Proposition 3.1.2 The probability of the impossible event equals zero, that is, $P(\emptyset) = 0$.

Proof. Since $P(\emptyset) = 1 - P(\Omega) = 0$. ■

As it was expected, sub-events (subsets) have smaller probabilities than events.

Proposition 3.1.3 If $A \subset B$ then $P(A) \leq P(B)$

Proof. The event B can be decomposed as the union of two disjoint events A and $A^c \cap B$, so we have that $P(B) = P(A) + P(A^c \cap B)$, that combined with the fact that $P(A^c \cap B) \geq 0$ proves the proposition. ■

As anticipated, sub-events (subsets) are associated with lesser probabilities than their corresponding events.

Proposition 3.1.4 Given that $A \subset B$, it follows that $P(A) \leq P(B)$.

Proof. The event B can be dissected into the union of two disjoint events A and $A^c \cap B$. Consequently, $P(B) = P(A) + P(A^c \cap B)$, which combined with the notion that $P(A^c \cap B) \geq 0$, substantiates the proposition. ■

With these fundamental elements established, we can demonstrate that probabilities range between zero and one.

Proposition 3.1.5 For each event A , $0 \leq P(A) \leq 1$.

Proof. By virtue of Axiom 1 and the consideration that $A \subset \Omega$ and consequently $P(A) \leq P(\Omega) = 1$. ■

Axiom 3 provides the means to compute the probability of the union of disjoint events, however, it does not extend to scenarios involving non-disjoint events. The succeeding proposition illustrates the method for computing the probability of the union of non-disjoint events.

Proposition 3.1.6 For any two events A and B , it follows that $P(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$.

Proof. The union of sets A and B can be represented as the union of two disjoint sets $A \cup B = B \cup (A \cap B^c)$. Given Axiom 3, we ascertain that

$$P(A \cup B) = P(B) + P(A \cap B^c)$$

Similarly, the set A can be deconstructed as the union of the disjoint sets $A = (A \cap B) \cup (A \cap B^c)$. As a result, we get

$$P(A \cup B^c) = P(A) - P(A \cap B)$$

The combination of both expressions yields the desired result. ■

The following equation extends to the scenario of n events A_1, \dots, A_n , employing the principle of inclusion-exclusion (see Section 2.3):

$$\begin{aligned} P\left(\bigcup_{i=1}^n A_i\right) &= \sum_{i=1}^n P(A_i) - \sum_{i < j} P(A_i \cap A_j) + \sum_{i < j < k} P(A_i \cap A_j \cap A_k) - \\ &\quad - \sum_{i < j < k < l} P(A_i \cap A_j \cap A_k \cap A_l) + \dots + (-1)^{n+1} P(A_1 \cap A_2 \cap \dots \cap A_n) \end{aligned}$$

A probability function is characterized as a function that assigns to every possible event within a sample space its corresponding probability.

Definition 3.1.3 Suppose (Ω, \mathcal{A}, P) denotes a probability space. A *probability function* is a real-valued function $f : \mathcal{A} \rightarrow [0, 1]$ such that for every $A \in \mathcal{A}$, $f(A) = P(A)$ holds true.

In Example 3.1, we introduced a probability space (Ω, \mathcal{A}, P) comprising n equally probable elements. The probability function associated with this experiment is defined as $f : \mathcal{A} \rightarrow [0, 1]$, such that $f(A) = d(A)/n$ for all $A \in \mathcal{A}$.

3.2 Conditional Probability

The principle of conditional probability is a cornerstone within the discipline of statistical learning. The conventional interpretation of conditional probability posits it as the recalibrated probability of event A following the occurrence of event B . This perspective, however, potentially implies a sequential or even causative linkage between events B and A , a suggestion which may not necessarily hold validity.

■ **Example 3.2** Suppose we are playing a game with a standard deck of 52 cards, and we draw two cards. Let event A be "drawing at least one heart" and event B be "drawing at least one queen". These two events are dependent since the occurrence of event B affects the probability of event A . However, these two events are not temporally related because the draw of the card happens at the same time - one event does not occur before the other. This example showcases the essence of dependency in probability theory without any temporal association between the events involved. ■

With reference to the axiomatization prescribed by Kolmogorov, conditional probability is initially introduced as a definitive construct. Certain scholars posit that, given its pivotal role within probability theory, conditional probability ought to be an attribute that is logically deduced from the foundational axioms. This perspective, naturally, necessitates an augmentation of Definition 3.1.2 with supplementary properties. Regrettably, there exists no agreed-upon method among mathematicians and philosophers regarding the manner in which this augmentation should be conducted.

Definition 3.2.1 Let A and B be two events such that $P(B) \neq 0$. The *conditional probability* of A given B , symbolized as $P(A | B)$, is elucidated as follows:

$$P(A | B) = \frac{P(A \cap B)}{P(B)}$$

By virtue of satisfying the axioms, a conditional probability is, in itself, a probability. The conditional probability $P(A | B)$ is undefined in instances where $P(B) = 0$.

The probability of two events transpiring concurrently (although not necessarily contemporaneously, as previously discussed in Example 3.2), given their respective conditional probabilities, is encapsulated by the formula $P(A \cap B) = P(A | B)P(B)$. This equation offers perhaps a more intuitive comprehension of the conditional probability concept. Indeed, there exist a number of authors who advocate for this interpretation to form the basis of the definition of conditional probability, as opposed to the quotient method.

The extension of this formula to accommodate n events, termed the *multiplication rule*, is expressed as follows:

$$P(A_1 \cap A_2 \cap \dots \cap A_n) = P(A_1)P(A_2 | A_1)\dots P(A_n | A_1 \cap A_2 \cap \dots \cap A_{n-1})$$

The notion of event independence holds significant importance in the realm of probability theory and statistical learning.

Definition 3.2.2 Two events A and B are declared to be *independent* if $P(A \cap B) = P(A)P(B)$.

From an intuitive perspective, the events A and B are considered independent if witnessing the occurrence of event B does not influence the probability of event A . This characteristic can be logically inferred from the definition of independence.

Proposition 3.2.1 Given two events A and B such that $P(A) > 0$ and $P(B) > 0$, A and B are independent if and only if $P(A | B) = P(A)$ and $P(B | A) = P(B)$.

Proof. Assume A and B are independent, implying that $P(A \cap B) = P(A)P(B)$. Then,

$$P(A | B) = \frac{P(A \cap B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

Proceeding from the assumption that $P(A | B) = P(A)$, and utilizing the multiplication rule, it follows that

$$P(A \cap B) = P(A | B)P(B) = P(A)P(B)$$

The same conclusion is drawn if the roles of A and B are interchanged. ■

Similar to the case of conditional probability, certain authors posit that independence, as a foundational concept in probability theory, ought to be a logical extension of the axioms, rather than being imposed as a definition.

The principle of independence can be expanded to accommodate multiple events: the events A_1, \dots, A_n are deemed to be independent (or mutually independent) if for every subset A_{i_1}, \dots, A_{i_j} comprising j events ($j = 2, 3, \dots, n$), it holds true that $P(A_{i_1} \cap \dots \cap A_{i_j}) = P(A_{i_1}) \dots P(A_{i_j})$.

■ **Example 3.3** A degree of confusion often arises regarding the distinction between mutually exclusive (or disjoint) events and independent events. For two mutually exclusive events A and B , the computation of the probability that A will transpire given B is somewhat nonsensical, since if B occurs, A is inherently impossible; analogously, discussing the conditional probability that A will occur given B when the probability of B is zero is likewise flawed. However, as Definition 3.2.2 does not explicitly exclude the instance of A and B being mutually exclusive, we are compelled to conclude that two mutually exclusive events are independent if, and only if, the probability of at least one (or both) of them is zero. ■

An intriguing scenario arises when events A and B are not independent, yet attain independence contingent on the occurrence of another event C .

Definition 3.2.3 Consider A , B and C as events such that $P(B \cap C) > 0$. A and B are considered *conditionally independent* given C if $P(A | B \cap C) = P(A | C)$.

■ **Example 3.4** Consider the act of rolling two dice; it is reasonable to assert that the outcomes of the two dice are independent from each other. That is, observing the outcome of one die provides no insight into the outcome of the other die. However, suppose the first die results in a four, and a third event is introduced - that the sum of the outcomes is an odd number - then this additional piece of information narrows the potential outcomes for the second die to only odd numbers. This illustrates the point that two events can be independent, yet fail to maintain conditional independence. ■

The ensuing theorem presents Bayes' rule, a fundamental principle underpinning a significant statistical learning technique known as Bayesian inference (see Section 7.1.1).

Theorem 3.2.2 (Bayes' Theorem) Let A and B represent two events with the condition that $P(B) \neq 0$. Consequently, we obtain that

$$P(A | B) = \frac{P(B | A)P(A)}{P(B)}$$

In this context, $P(A)$ is referred to as the *prior probability*, while $P(A | B)$ is deemed the *posterior probability*.

Proof. As per the definition of conditional probability, $P(A | B) = P(A \cap B) / P(B)$ (given $P(B) \neq 0$) and $P(B | A) = P(A \cap B) / P(A)$ (provided $P(A) \neq 0$). By solving for $P(A \cap B)$ and substituting into the previous expressions for $P(A | B)$, we arrive at the theorem. ■

As evident in the proof, Bayes' theorem is a direct derivative of the definition of conditional probability, notwithstanding our misgivings about conditional probability being a definition.

According to Bayesian inference, it facilitates the computation of how our degree of certainty about event A (the prior probability $P(A)$) evolves when we acquire supplementary evidence via the occurrence of event B (transforming into the posterior probability $P(A | B)$).

■ **Example 3.5** Consider E to be a disease affecting one in every million people, $P(E) = 1 \times 10^{-6}$, and let $+$ represent a test devised to detect the disease, with a failure rate of one in every thousand applications, $P(+ | E) = 999/1000$. We aim to determine the probability of disease presence if the test is positive $P(E | +)$. Upon employing Bayes' theorem, we find that:

$$P(E | +) = \frac{P(+ | E)P(E)}{P(+)} = \frac{P(+ | E)P(E)}{P(+ | E)P(E) + P(+ | E^c)P(E^c)} = 0.001$$

This implies that despite the test only failing once per thousand applications, it remains highly improbable that we have the disease following a positive result. This paradoxical outcome can be attributed to the higher probability of test failure 10^{-3} compared to the likelihood of disease occurrence 10^{-6} . Practically, this issue is circumvented by applying a second test to individuals who received a positive result, as the probability of disease presence following two positive results is 0.5 (under the assumption that the successive test repetitions are independent). ■

Bayes' theorem is most useful when the events involved are dependent and when new information about one event can update our understanding of the other event's probability.

■ **Example 3.6** Suppose you're drawing a single card from a standard deck of 52 playing cards. Let Event A be "drawing a red card" and Event B be "drawing a queen". In this context, the use of Bayes' theorem to compute $P(A | B)$, the probability of drawing a red card given that a queen has been drawn, would not yield a meaningful result because the event B provides no new information that would affect the probability of event A. ■

Bayes' theorem can indeed be extended to accommodate multiple events. Consider a set of events A_1, \dots, A_k such that $P(A_j) > 0$ for all j in the range of 1 to k . Suppose these events constitute a partition of the sample space Ω . Now, let B denote an event with the property that $P(B) > 0$. In such a context, it can be deduced for each i in the range of 1 to k that the conditional probability $P(A_i | B)$ is given by the formula

$$P(A_i | B) = \frac{P(B | A_i) P(A_i)}{\sum_{j=1}^k P(B | A_j) P(A_j)}$$

This illustrates the capacity of Bayes' theorem to apply to a broader set of scenarios involving multiple events.

3.3 Random Variables

A random variable is a function that assigns a real number to each possible outcome of an experiment. Therefore, it serves as a quantitative representation of the results of the experiment. Random variables are very useful since they offer a quantifiable means to examine the outcomes of the experiments and to discover their analytical properties. Such is the efficacy of random variables that a majority of statisticians primarily consider their investigations within the framework of random variables as opposed to probability spaces.

Definition 3.3.1 Let (Ω, \mathcal{A}, P) be a discrete probability space. A *discrete random variable* is a function $X : \Omega \rightarrow \mathbb{R}$ mapping from the sample space Ω to the real numbers.

The terminology "random variable" might potentially lead to some confusion. Firstly, these are not variables in the conventional algebraic sense, but rather, they are functions. Secondly, they are not inherently random; it is the experiment that they represent which possesses randomness. Despite these points of potential confusion, we adhere to the established terminology.

Random variables hold increased utility when they encapsulate properties of the experiment. For instance, if the sample space consists of a school's student body, a random variable could associate each student with their respective height. Random variables also enable us to redistribute elements of the sample space into new events. For example, if two dice are rolled, a random variable could represent the sum of the dice's outcomes. It is crucial to remember that we possess the liberty to assign a random variable to any sample space, even when the assignment might not seem intuitively meaningful. As an illustration, one could assign a numerical value to each possible color in a deck of cards, draw two cards randomly, and sum the assigned numbers of these two cards. Although such a setup may not yield a significant interpretation, it is nonetheless possible to calculate an array of probabilities based on this setup.

Definition 3.3.2 Let $X : \Omega \rightarrow \mathbb{R}$ be a discrete random variable, and let $\{x_1, x_2, \dots, x_i, \dots\}$ be the range of X . The probability of X being equal to x_i , expressed as $P(X = x_i)$, is given by $P(X = x_i) = P(\{\omega \in \Omega : X(\omega) = x_i\})$. Let $C \subset \mathbb{R}$ be a subset such that the set $\{\omega \in \Omega : X(\omega) \in C\}$ constitutes an event. The probability of X belonging to C , expressed as $P(X \in C)$, is given by $P(X \in C) = P(\{\omega \in \Omega : X(\omega) \in C\})$.

The probability of a random variable X essentially configures a probability space over the line of real numbers, specifically over the range of X .

■ **Example 3.7** Let $\Omega = \{1, 2, 3, 4, 5, 6\}$ be the sample space of possible outcomes when tossing a die, $\mathcal{A} = \mathcal{P}(\Omega)$ be all possible events of Ω , and P a probability that assigns $1/6$ to each single outcome in Ω . Let $X : \Omega \rightarrow \mathbb{R}$ be a random variable defined as:

$$X(\omega) = \begin{cases} 0 & \text{if } \omega \text{ is even (2, 4, 6),} \\ 1 & \text{if } \omega \text{ is odd (1, 3, 5).} \end{cases}$$

This random variable maps the outcomes of the die toss to either 0 (if the outcome is even) or 1 (if the outcome is odd). For $C = \{0\}$, the probability $P(X \in C) = P(X = 0) = P(\{2, 4, 6\}) = 1/2$. For $C = \{1\}$ the probability $P(X \in C) = P(X = 1) = P(\{1, 3, 5\}) = 1/2$. Through this transformation, the original probability space has been mapped to the real numbers using the random variable X , establishing a new probability over X 's range. ■

Definition 3.1.3 introduced the concept of probability function for discrete probability spaces based on the probabilities of the events. Next definition extends the concept of probability function to discrete random variables.

Definition 3.3.3 Let X be a random variable over a discrete probability space, and let $\{x_1, x_2, \dots\}$ be the range of X . The *probability function* of the random variable X , abbreviated as p.f., is defined as the function $f : \text{range}(X) \rightarrow [0, 1]$ such that $f(x_i) = P(X = x_i)$.

Of course, the probability function is defined only if $\{\omega \in \Omega : X(\omega) = x_i\}$ is an event. Unless we say the contrary, and since we are dealing mostly with discrete probability spaces, we will assume that $\{\omega \in \Omega : X(\omega) = x_i\}$ is an event for all the points that compose the range of X . The set of points for which the probability function is greater than zero, that is $\{x : f(x) > 0\}$, is called the *support* of the distribution of X .

It is possible for two random variables to have identical probability functions but to differ in significant ways.

■ **Example 3.8** Let $\Omega = \{H, T\}$ be the sample space of possible outcomes when tossing a coin, and P a probability that assigns $1/2$ to each single outcome in Ω . Let $X : \Omega \rightarrow \mathbb{R}$ be a random variable defined as: X such that $X = 1$ if the coin shows Head and $X = 0$ if coin shows Tail. The individual probability distributions of X is $P(X = 1) = P(Y = 1) = 0.5$. The random variables of this example and Example 3.7 have same probability distribution even if they are different random variables. ■

Given the probability function of a random variable, we can derive the probability of any subset of the real line.

Proposition 3.3.1 Let X be a discrete random variable with probability function f . The probability of each subset C of the real line can be determined from the relation $P(X \in C) = \sum_{x_i \in C} f(x_i)$

Proof. Considering that each outcome in the sample space is associated with exactly one value in the range $\{x_1, x_2, \dots, x_i, \dots\}$ of X , we have that:

$$P(X \in C) = P(\{\omega \in \Omega : X(\omega) \in C\}) = \sum_{x_i \in C} P(X = x_i) = \sum_{x_i \in C} f(x_i)$$

■

Next proposition outlines a fundamental property, that the total sum of probabilities across all possible outcomes of a discrete random variable is 1.

Proposition 3.3.2 Let X be a discrete random variable with probability function f . If $\{x_1, x_2, \dots\}$ is the range of X , then $\sum_{i=1}^{\infty} f(x_i) = 1$.

Proof. Considering that X is a total function, that each outcome in the sample space is associated with exactly one value in the range $\{x_1, x_2, \dots\}$, and given the axiomatic definition of probability we have that:

$$\begin{aligned}\sum_{i=1}^{\infty} f(x_i) &= f(x_1) + f(x_2) + \dots = P(X = x_1) + P(X = x_2) + \dots = \\ &= P(\{\omega \in \Omega : X(\omega) = x_1\}) + P(\{\omega \in \Omega : X(\omega) = x_2\}) + \dots = 1\end{aligned}$$

■

The cumulative distribution function represents the probability that a random variable takes on a value less than or equal to a specific point.

Definition 3.3.4 The *cumulative distribution function* (abbreviated c.d.f.) F of a random variable X is the function $F(x) = Pr(X \leq x)$ for all $-\infty < x < \infty$

If X follows a discrete distribution characterized by the probability function $f(x)$, its cumulative distribution function $F(x)$ will exhibit the following behavior: at each distinct value x_i of X , $F(x)$ will display a jump equal to $f(x_i)$; between these distinct values, $F(x)$ remains unchanged.

The cumulative distribution function allows us to see how probabilities accumulate over the range of a random variable, offering insights into the overall distribution of the data.

■ **Example 3.9** Let's X be a discrete random variable that represents the grades of students in a class. Each grade is between 0 and 10. The probability that a student receives a grade of x is given by the probability mass function $p(x)$. The cumulative distribution function represents the probability that a randomly selected student scores x or less. For example, a value of $F(7) = 0.6$ would mean that there's a 60% chance a student picked at random scored 7 or below. ■

The cumulative distribution function of a random variable is non-decreasing.

Proposition 3.3.3 Let $F(X)$ be the cumulative distribution function of a random variable X . Then, if $x_1 < x_2$ we have that $F(x_1) \leq F(x_2)$.

Proof. Given two values x_1 and x_2 where $x_1 < x_2$, the set of outcomes where $X \leq x_1$ is a subset of the outcomes where $X \leq x_2$. Therefore, the probability of X taking a value less than or equal to x_1 will be less than or equal to the probability of X taking a value less than or equal to x_2 . Then $F(x_1) \leq F(x_2)$. ■

Next proposition delineates the asymptotic properties of the cumulative distribution function of a random variable, showcasing its bounds as we approach negative and positive infinity.

Proposition 3.3.4 Let $F(X)$ be the cumulative distribution function of a random variable X . Then, we have that $\lim_{x \rightarrow -\infty} F(x) = 0$ and that $\lim_{x \rightarrow \infty} F(x) = 1$.

Proof. As x tends to negative infinity, the probability that the random variable X takes on a value less than or equal to this increasingly smaller x tends to zero. This is because there are fewer and

fewer values (or none, depending on the specifics of the distribution) that X can assume which are less than this increasingly negative x . Therefore:

$$\lim_{x \rightarrow -\infty} F(x) = \lim_{x \rightarrow -\infty} P(X \leq x) = 0$$

As x tends to positive infinity, the probability that the random variable X takes on a value less than or equal to this increasingly larger x approaches 1. This is because, given the unbounded increase of x , it encapsulates all possible values that X can take on. Therefore:

$$\lim_{x \rightarrow \infty} F(x) = \lim_{x \rightarrow \infty} P(X \leq x) = 1$$

■

The probability of X exceeding x is given by the complement of the cumulative distribution function at that point.

Proposition 3.3.5 Let $F(X)$ be the cumulative distribution function of a random variable X . Then, for every $x \in X$ we have that $P(X > x) = 1 - F(x)$.

Proof. The probability that X takes on a value greater than x plus the probability that X takes on a value less than or equal to x should sum up to 1. Given this, the probability that X takes a value greater than x is:

$$P(X > x) = 1 - P(X \leq x)$$

Using the definition of the cumulative distribution function, we get:

$$P(X > x) = 1 - F(x)$$

■

The following proposition establishes a relationship between the probabilities of a random variable X falling between two specific values and the corresponding differences in its cumulative distribution function values at those points.

Proposition 3.3.6 Let $F(X)$ be the cumulative distribution function of a random variable X . Then, for all values $x_1, x_2 \in X$ such that $x_1 < x_2$ we have that $P(x_1 < X \leq x_2) = F(x_2) - F(x_1)$

Proof. The probability that X is less than or equal to x_2 is $F(x_2)$. From this, if we subtract the probability that X is less than or equal to x_1 , which is $F(x_1)$, we'll get the probability that X falls strictly between x_1 and x_2 :

$$P(x_1 < X \leq x_2) = F(x_2) - F(x_1)$$

■

3.3.1 Multivariate Distributions

Multivariate distributions show comparisons between two or more measurements and the relationships among them [...] For discrete random variables, multivariate distribution and described by joint probabilities.

We will start by introducing bivariate distributions and studying their properties, and then we will generalize to the case of multivariate distributions. A bivariate distribution is the simplest form of multivariate distribution, it is comprised by a pair of random variables.

Definition 3.3.5 Let $X_1 : \Omega_1 \rightarrow \mathbb{R}$ and $X_2 : \Omega_2 \rightarrow \mathbb{R}$ be two random variables. The *joint probability distribution* or *bivariate probability distribution* of X_1 and X_2 is defined as the collection of all probabilities of the form $P((X_1, X_2) \in C)$ for all sets $C \subset \mathbb{R} \times \mathbb{R}$ of pairs of real numbers such that $((\omega_1, \omega_2) \in \Omega_1 \times \Omega_2 : (X_1(\omega_1), X_2(\omega_2)) \in C)$ is an event.

The joint probability distribution of two random variables X_1 and X_2 defines a probability space in \mathbb{R}^2 . If the random variables X_1 and X_2 each have a discrete distribution, then the joint distribution is also a discrete distribution.

Definition 3.3.6 Let X_1 and X_2 be two random variables over discrete probability spaces. The *joint probability mass function* of the random variables X_1 and X_2 is defined as the function $f : \text{range}(X_1) \times \text{range}(X_2) \rightarrow [0, 1]$ such that $f(x_1, x_2) = P(X_1 = x_1, X_2 = x_2)$.

Proposition 3.3.7 Let X and Y have a discrete joint distribution. If (x, y) is not one of the possible values of the pair (X, Y) , then $f(x, y) = 0$. Also, $\sum_{(x,y)} f(x, y) = 1$

Proof. ■

Finally, for each C of ordered pairs $P[(X, Y) \in C] = \sum_{(x,y) \in C} f(x, y)$

Any function that satisfied the two displayed formulas is the joint p.d.f. for some probability distribution.

A particular interesting case of bivariate distribution is given by the sum of two random variables. This is a highly confusing scenario, since we are not adding two probability distributions, as the notation $X + Y$ might suggest. Instead, we are defining a new random variable over the cartesian product of the original sample spaces.

Definition 3.3.7 Let $X : \Omega_X \rightarrow \mathbb{R}$ and $Y : \Omega_Y \rightarrow \mathbb{R}$ be two random variables. The sum distribution of X and Y , denoted by $X + Y$, is defined as the random variable $X + Y : \Omega_X \times \Omega_Y \rightarrow \mathbb{R}$ that assigns to each pair $(a, b) \in \Omega_X \times \Omega_Y$ the number $X(a) + Y(b)$.

TODO: Generalization to n random variables.

3.3.2 Marginal Distribution

Often, we start with a joint distribution of two random variables and we then want to find the distribution of just one of them, called the *marginal distribution*.

Definition 3.3.8 Suppose that X and Y have a joint distribution. The c.d.f. of X derived with the above theorem is called the marginal c.d.f. of X . Similarly, the p.f. or p.d.f. of X associated with the marginal c.d.f. of X is caled the marginal p.f. or marginal p.d.f. of X .

Proposition 3.3.8 If X and Y have a discrete joint distribution for which the joint p.f. is f , then the marginal p.f. of X is $f_1(x) = \sum_y f(x, y)$

Proof. ■

Similarly, the marginal p.f. f_2 of Y is $f_2(y) = \sum_x f(x, y)$.

Although the marginal distributions of X and Y can be derived from their joint distribution, it is not possible to reconstruct the joint distribution of X and Y from their marginal distributions without additional information.

Independent Random Variables

Definition 3.3.9 It is said that two random variables X and Y are independent if, for every two sets A and B of real numbers such that $\{X \in A\}$ and $\{Y \in B\}$ are events $P(X \in A \text{ and } Y \in B) = Pr(X \in A)Pr(Y \in B)$

Proposition 3.3.9 Suppose that X and Y are random variables that have a joint p.f., p.d.f., or p.f./p.d.f. f . Then X and Y will be independent if and only if f can be represented in the following form for $-\infty < x < \infty$ and $-\infty < y < \infty$ $f(x, y) = h_1(x)h_2(y)$ where h_1 is a nonnegative function of x alone and h_2 is a nonnegative function of y alone.

Proof.

Corollary 3.3.10 Two random variables X and Y are independent if and only if the following factorization is satisfied for all real numbers x and y $f(x, y) = f_1(x)f_2(y)$

Proof.

Two discrete random variables X and Y are independent if, for each y , learning that $Y = y$ does not change any of the probabilities of the events $\{X = x\}$.

If X and Y are independent, then $h(X)$ and $g(Y)$ are independent no matter what the functions h and g are.

3.3.3 Conditional Distributions

The conditional distribution of one random variable X given another Y is the distribution that we would use for X after we learn the value of Y .

Definition 3.3.10 Let X and Y have a discrete joint distribution with joint p.f. f . Let f_2 denote the marginal p.f. of Y . For each y such that $f_2(y) > 0$, define $g_1(x | y) = \frac{f(x, y)}{f_2(y)}$

Then g_1 is called the conditional p.f. of X given Y . The discrete distribution whose p.f. is $g_1(\cdot | y)$ is called the conditional distribution of X given that $Y = y$.

Construction of the Joint Distribution

Proposition 3.3.11 Let X and Y be random variables such that X has p.f. or p.d.f. $f_1(x)$ and Y has p.f. or p.d.f. $f_2(y)$. Also, assume that the conditional p.f. or p.d.f. of X given $Y = y$ is $g_1(x | y)$ while the conditional p.f. or p.d.f. of Y given $X = x$ is $g_2(y | x)$. Then for each y such that $f_2(y) > 0$ and each x , $f(x, y) = g_1(x | y)f_2(y)$ where f is the joint p.f., p.d.f. or p.f./p.d.f. of X and Y . Similarly, for each x such that $f_1(x) > 0$ and each y , $f(x, y) = f_1(x)g_2(y | x)$

Proof.

Next theorem provides a generalization of the law of total probability to random variables.

Proposition 3.3.12 If $f_2(y)$ is the marginal p.f. or p.d.f. of a random variable Y and $g_1(x | y)$ is the conditional p.f. or p.d.f. of X given $Y = y$, then the marginal p.f. or p.d.f. of X is $f_1(x) = \sum_y g_1(x | y)f_2(y)$ if Y is discrete.

Proof.

The following theorem is a generalization of Bayes' theorem for random variables.

Theorem 3.3.13 If $f_2(y)$ is the marginal p.f. or p.d.f. of a random variable Y and $g_1(x | y)$ is the conditional p.f. or p.d.f. of X given $Y = y$, then the conditional p.f. or p.d.f. of Y given $X = x$ is $g_2(y | x) = \frac{g_1(x | y)f_2(y)}{f_1(x)}$

Proof.

Proposition 3.3.14 Suppose that X and Y are two random variables having a joint p.f., p.d.f., or p.f./p.d.f. f . Then X and Y are independent if and only if for every value of y such that $f_2(y) > 0$ and every value of x $g_1(x | y) = f_1(x)$

Proof. ■

Conclude that all the above can be generalized to multivariate distributions, and provide a couple of examples of such generalization.

3.4 Random Samples

Perhaps this concept should be moved to the section of statistical inference

The concept of random sample plays a very important role in the area of statistical learning. Assuming that a collection of random variables form a random sample simplifies the mathematics behind inference methods. However, the requirements behind random samples are not always satisfied in practice.

Definition 3.4.1 Let f be a probability distribution, and let X_1, X_2, \dots, X_n be n random variables. It is said that the variables X_1, X_2, \dots, X_n form a *random sample* for the distribution f if they are independent, and the marginal distribution of each of them is f .

If the random variables X_1, X_2, \dots, X_n form a random sample for the distribution f , it is said that they are *independent and identically distributed*, abbreviated *i.i.d.* The number n is called the *sample size*. The joint distribution g of the random sample is given by:

$$g(x_1, x_2, \dots, x_n) = f(x_1) f(x_2) \dots f(x_n)$$

for all the points $(x_1, x_2, \dots, x_n) \in \mathcal{R}$.

3.5 Characterizing Distributions

A *measure of central tendency* is a number derived from a probability distribution, intended as a summary of that distribution. The most common measures of central tendency in use are the mean and the median. Each of these measures provides a different approach to characterize distributions. It is also common to use *metrics of dispersion* to describe the variability of a distribution around the measures of centrality. We will review two metrics of dispersion, the variance and the standard deviation. The metrics of dispersion can also be used in case of bivariate distributions, under the names of covariance and correlation, to measure the *statistical relationship* between two random variables. All these measures allow us to summarize and compare distributions.

3.5.1 Measures of Central Tendency

The most commonly used measure of central tendency is the *mean*. The mean of a collection of outcomes is the weighted average of these outcomes, where the weights are equal to the probabilities. The mean is also known as expected value or expectation.

Definition 3.5.1 Let X be a discrete random variable whose probability function is f . The *mean* of X , denoted by $E(X)$, is defined as:

$$E(X) = \sum_x x f(x)$$

Of course, Definition 3.5.1 only makes sense if the summation converges. It is also possible that the mean is infinite, but in this book we are only interested in finite means. We have defined the

concept of mean based on random variables. In this sense, the definition of mean only takes into account the distribution of the random variables, not the original outcomes. That is, two different random variables with the same distribution will have the same mean. When working with random variables it is common to use the name expected value instead of mean. However, this name is a little bit misleading, since for the majority of the discrete distributions, the expected value is not one of the possible values of the distribution, i.e., the expected value is not expected at all. For example, if we throw a dice, the expected value would be 3.5. This undesired property of something known as expected value has generated a lot of confusion in scientific research. The expectation of a random variable has a physical interpretation as the center of gravity of the distribution. Expectation, as the center of gravity, can be greatly affected by a small change in the probability assigned to a large value of X .

The expectation of the linear combination of n random variables is the linear combination of their expectations.

Proposition 3.5.1 Let X_1, \dots, X_n be n independent discrete random variables with expectations $E(X_i)$, and let a_1, \dots, a_n and b constants, then

$$E(a_1X_1 + \dots + a_nX_n + b) = a_1E(X_1) + \dots + a_nE(X_n) + b$$

Proof. We have that

$$\begin{aligned} E(a_1X_1 + \dots + a_nX_n + b) &= \sum_{x_1} \dots \sum_{x_n} (a_1x_1 + \dots + a_nx_n + b) f(x_1, \dots, x_n) = \\ &= \sum_{x_1} \dots \sum_{x_n} a_1x_1 f(x_1, \dots, x_n) + \dots + \sum_{x_1} \dots \sum_{x_n} a_nx_n f(x_1, \dots, x_n) + \sum_{x_1} \dots \sum_{x_n} b f(x_1, \dots, x_n) = \\ &= \sum_{x_1} a_1x_1 f(x_1) + \dots + \sum_{x_n} a_nx_n f(x_n) + b = a_1 \sum_{x_1} x_1 f(x_1) + \dots + a_n \sum_{x_n} x_n f(x_n) + b = \\ &= a_1E(X_1) + \dots + a_nE(X_n) + b \quad (3.1) \end{aligned}$$

■

The expectation of the product of n independent random variables is the product of the individual expectations.

Proposition 3.5.2 Let X_1, \dots, X_n be n independent discrete random variables with expectations $E(X_i)$, then:

$$E\left(\prod_{i=1}^n X_i\right) = \prod_{i=1}^n E(X_i)$$

Proof. We have that

$$\begin{aligned} E(X_1 \cdot \dots \cdot X_n) &= \sum_{x_1} \dots \sum_{x_n} (x_1 \cdot \dots \cdot x_n) f(x_1, \dots, x_n) = \\ &= \sum_{x_1} \dots \sum_{x_n} x_1 f(x_1, \dots, x_n) \cdot \dots \cdot \sum_{x_1} \dots \sum_{x_n} x_n f(x_1, \dots, x_n) = \\ &= \sum_{x_1} x_1 f(x_1) \cdot \dots \cdot \sum_{x_n} x_n f(x_n) = E(X_1) \cdot \dots \cdot E(X_n) \quad (3.2) \end{aligned}$$

■

The expectation of the product of non-independent random variables is not necessarily equal to the product of their individual expectations.

In the area of statistical inference it is also highly convenient to compute the sample mean, as the average of n random variables. In particular, we will compute the sample mean of random samples.

Definition 3.5.2 Let X_1, X_2, \dots, X_n be n random variables. The *sample mean*, denoted by \bar{X}_n , is defined as:

$$\bar{X}_n = \frac{1}{n} \sum_{i=1}^n X_i$$

Do not confuse $\frac{1}{n}(X_1 + X_2 + \dots + X_n)$, which is a probability distribution, with $E(X_1 + X_2 + \dots + X_n)$, which is a real number.

The Median

We have seen that the mean of a probability distribution is the center of gravity of that distribution. The actual center of the distribution is called the *median*.

Definition 3.5.3 Let X be a discrete random variable. Every number m that satisfy the following properties is called a median of the distribution of X :

$$Pr(X \leq m) \geq 1/2 \quad \text{and} \quad Pr(X \geq m) \geq 1/2$$

The median divides a probability distribution in two equal parts. A distribution could have more than one median. And, on the contrary of what happens in case of the expectation, every distribution must have at least one median. An advantage of the median over the mean is that we can move a value x larger to the median to any arbitrary larger value, and the median will remain the same.

3.5.2 Measures of Dispersion

Definitions of the Variance and the Standard Deviation

Definition 3.5.4 Let X be a random variable with finite mean and $\mu = E(X)$. The variance of X , denoted by $Var(X)$, is defined as follows: $Var(X) = E[(X - \mu)^2]$

If X has infinite mean or if the mean of X does not exist, we say that $Var(X)$ does not exist. The standard deviation of X is the nonnegative square root of $Var(X)$ if the variance exists. It is common to denote the standard deviation by the symbol σ , and the variance by σ^2 . Variance depends only on the distribution.

Proposition 3.5.3 For every random variable X , $Var(X) = E(X^2) - [E(X)]^2$.

Proof.

The variance (as well as the standard deviation) of a distribution provides a measure of the spread or dispersion of the distribution around its mean μ . The variance of a distribution, as well as its mean, can be made arbitrarily large by placing even a very small but positive amount of probability far enough from the origin on the real line.

Properties of the Variance

Proposition 3.5.4 For constants a and b , let $Y = aX + b$, then $Var(Y) = a^2Var(X)$ and $\sigma_Y = |a|\sigma_X$.

Proof.

If X_1, \dots, X_n are independent random variables with finite means, and if a_1, \dots, a_n and b are arbitrary constants, then $Var(a_1X_1 + \dots + a_nX_n + b) = a_1^2Var(X_1) + \dots + a_n^2Var(X_n)$

■ Example 3.10 The Variance of a Binomial Distribution

The variance of a random variable X with a binomial distribution of n samples with probability p is $Var(X) = np(1 - p)$

3.5.3 Measures of Statistical Relationship

Covariance and correlation are attempts to measure the linear dependence between two random variables.

Covariance

Definition 3.5.5 Definition 183. Let X and Y be random variables having finite means. Let $E(X) = \mu_X$ and $E(Y) = \mu_Y$. The covariance of X and Y , which is denoted by $\text{Cov}(X, Y)$ is defined as $\text{Cov}(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$

if the expectation exists.

The covariance between X and Y is intended to measure the degree to which X and Y tend to be large at the same time or the degree to which one tends to be large while the other is small.

Proposition 3.5.5 For all random variables X and Y such that $\sigma_X^2 < \infty$ and $\sigma_Y^2 < \infty$ $\text{Cov}(X, Y) = E(XY) - E(X)E(Y)$

Proof. ■

Correlation

Correlation is a measure of association between two random variables that is not driven by arbitrary changes in the scales.

Definition 3.5.6 Let X and Y be random variables with finite variances σ_X^2 and σ_Y^2 respectively. Then the correlation of X and Y , which is denoted by $\rho(X, Y)$, is defined as follows $\rho(X, Y) = \frac{\text{Cov}(X, Y)}{\sigma_X \sigma_Y}$

XX

Definition 3.5.7 It is said that X and Y are positively correlated if $\rho(X, Y) > 0$, that X and Y are negatively correlated if $\rho(X, Y) < 0$ and that X and Y are uncorrelated if $\rho(X, Y) = 0$.

Properties of Covariance and Correlation

Proposition 3.5.6 Moreover $-1 \leq \rho(X, Y) \leq 1$

Proof. ■

Proposition 3.5.7 If X and Y are independent random variables with $0 < \sigma_X^2 < \infty$ and $0 < \sigma_Y^2 < \infty$ then $\text{Cov}(X, Y) = \rho(X, Y) = 0$

Proof. ■

The converse is not true as a general rule. Two dependent random variables can be uncorrelated.

Proposition 3.5.8 Suppose that X is a random variable such that $0 < \sigma_X^2 < \infty$ and $Y = aX + b$ for some constants a and b , where $a \neq 0$. If $a > 0$ then $\rho(X, Y) = 1$. If $a < 0$, then $\rho(X, Y) = -1$.

Proof. ■

The converse is also true, that is, if $|\rho(X, Y)| = 1$ implies that X and Y are linearly related.

Proposition 3.5.9 If X and Y are random variables such that $\text{Var}(X) < \infty$ and $\text{Var}(Y) < \infty$, then $\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y)$

Proof. ■

For all constants a and b , it can be shown that $\text{Cov}(aX, bY) = ab\text{Cov}(X, Y)$.

Proposition 3.5.10 Let X and Y are random variables such that $\text{Var}(X) < \infty$ and $\text{Var}(Y) < \infty$, and let a , b and c be constants, then $\text{Var}(aX + bY + c) = a^2\text{Var}(X) + b^2\text{Var}(Y) + 2ab\text{Cov}(X, Y)$

Proof. ■

A special case is $\text{Var}(X - Y) = \text{Var}(X) + \text{Var}(Y) - 2\text{Cov}(X, Y)$

Proposition 3.5.11 If X_1, \dots, X_n are random variables such that $\text{Var}(X_i) < \infty$ for $i = 1, \dots, n$, then $\text{Var}(\sum_{i=1}^n X_i) = \sum_{i=1}^n \text{Var}(X_i) + 2 \sum \sum \text{Cov}(X_i, X_j)$

Proof. ■

Proposition 3.5.12 If X_1, \dots, X_n are uncorrelated random variables, then $\text{Var}(\sum_{i=1}^n X_i) = \sum_{i=1}^n \text{Var}(X_i)$

Proof. ■

3.6 Common Distributions

3.6.1 Uniform Distribution

Definition 3.6.1 Let $a \leq b$ be integers. Suppose that the value of a random variable X is equally likely to be each of the integers a, \dots, b . Then we say that X has the uniform distribution on the integers a, \dots, b .

Introduce the following proposition

Proposition 3.6.1 If X has the uniform distribution on the integers a, \dots, b , the p.f. of X is

$$f(x) = \begin{cases} \frac{1}{b-a+1} & \text{for } x = a, \dots, b \\ 0 & \text{otherwise} \end{cases}$$

Proof. ■

The uniform distribution on the integers a, \dots, b represents the outcome of an experiment that is often described by saying that one of the integers a, \dots, b is chosen at random. A uniform distribution cannot be assigned to an infinite sequence of possible values

3.6.2 Bernoulli Distributions

■ **Example 3.11** A random variable Z that takes only two values 0 and 1 with $P(Z = 1) = p$ has the Bernoulli distribution with parameter p . We also say that Z is a Bernoulli random variable with parameter p . ■

3.6.3 Binomial Distributions

■ **Example 3.12** Suppose we perform N independent trials where each trial either succeeds or fails with probability of success p , and let X the random variable defined by the number of successes. The probability of having exactly n successes $P(X = n)$ follows a *binomial distribution* with parameters N, p , defined by:

$$f(n | N, p) = \binom{N}{n} p^n (1-p)^{(N-n)}$$

Definition 3.6.2 The discrete distribution represented by the p.f.f $(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & \text{for } x = 0, 1, \dots, n \\ 0 & \text{otherwise} \end{cases}$

is called the binomial distribution with parameters n and p .

Consider a general experiment that consists of observing n independent trials with only two possible results for each trial: success and failure. Then the distribution of the number of trials that result in success will be binomial with parameters n and p , where p is the probability of success on each trial.

3.7 Large Random Samples

The law of large numbers provides the mathematical foundation to the intuition that the average of a large sample of independent and identically distributed random variables should be close to their mean. The central limit theorem is a practical method that allow us to approximate the probability that the sample average is close to the true mean.

Write a more elaborated introduction to the section.

3.7.1 Law of Large Numbers

The *law of large numbers* is a theorem that states that the average of a large sample of independent and identically distributed random variables should be close to their mean, and that the more variables we add, the closer will be to that value. In practice, the law of large numbers allow us to describe the expected value of performing the same experiment a large number of times.

In this section we are going to prove the weak version of law of large numbers. Explain the difference between the weak and the strong versions of the law.

Before to prove the law of large numbers we need to prove two other related propositions: Markov's inequality and Chebyshev's inequality. Markov's inequality puts a bound on how much arbitrarily large can be the values of a nonnegative random variable given its mean.

Proposition 3.7.1 — Markov's Inequality. Let X be a nonnegative random variable, i.e. $P(X \geq 0) = 1$ with mean μ . Then for every real number $t > 0$ we have that

$$P(X \geq t) \leq \frac{\mu}{t}$$

Proof. Let f be the probability mass function of X . The mean μ of X is given by $\mu = \sum_x x f(x)$, but since X is non-negative we have that $\mu = \sum_{x>0} x f(x)$. Then

$$\mu = \sum_{x>0} x f(x) = \sum_{x=0}^t x f(x) + \sum_{x=t}^{\infty} x f(x) \geq \sum_{x=t}^{\infty} x f(x) \geq \sum_{x=t}^{\infty} t f(x) = t \sum_{x=t}^{\infty} f(x) = t P(X \geq t)$$

that is, $\mu \geq t P(X \geq t)$. ■

Chebyshev's inequality uses the variance to bound on how much arbitrarily large can be the values of the random variable given its mean. Chebyshev's inequality does not require the random variable to be nonnegative.

Corollary 3.7.2 — Chebyshev's inequality. Let X be a random variable with mean μ and variance σ^2 . Then for every real number $t > 0$ we have that

$$P(|X - \mu| \geq t) \leq \frac{\sigma^2}{t^2}$$

Proof. Applying Markov's inequality we have that

$$P(|X - \mu| \geq t) = P((X - \mu)^2 \geq t^2) \leq \frac{E((X - \mu)^2)}{t^2} = \frac{\sigma^2}{t^2}$$

■

The last element we need to formally prove the law of large numbers is to introduce the concept of convergence in probability for random variables. A sequence of random variables converges in probability to a value b if X_n lies in all the intervals around b .

Definition 3.7.1 Let X_1, X_2, \dots be a sequence of random variables. It is said that the sequence X_1, X_2, \dots converges in probability to b , denoted by $X_n \xrightarrow{P} b$, if for every positive real number $\varepsilon > 0$ we have that

$$\lim_{n \rightarrow \infty} P(|X_n - b| < \varepsilon) = 1$$

Given the above definitions and partial results, we can now formally introduce and prove the law of large numbers.

Theorem 3.7.3 — Law of Large Numbers. Let X_1, \dots, X_n be a random sample with finite mean $E(X_i) = \mu < \infty$ for all i and finite variance $Var(X_i) = \sigma^2 < \infty$ for all i , and let $\bar{X}_n = \frac{1}{n}(X_1 + \dots + X_n)$ be the sample mean. Then we have that

$$\bar{X}_n \xrightarrow{P} \mu$$

Proof. Since the variables X_1, \dots, X_n are independent and identically distributed, we have that the variance of the sample mean is $Var(\bar{X}_n) = \frac{\sigma^2}{n}$ and that the mean is $E(\bar{X}_n) = \mu$ (see XX). Applying the Chebyshev's inequality to the random variable \bar{X}_n we have that

$$P(|\bar{X}_n - \mu| \geq \varepsilon) \leq \frac{\sigma^2}{n\varepsilon^2}.$$

From there we can obtain

$$P(|\bar{X}_n - \mu| < \varepsilon) = 1 - P(|\bar{X}_n - \mu| \geq \varepsilon) \geq 1 - \frac{\sigma^2}{n\varepsilon^2}.$$

As n approaches infinity, the above expression approaches 1. Applying the definition of convergence in probability, we have that

$$\bar{X}_n \xrightarrow{P} \mu$$

■

It is very important to note that the law of large numbers only works in case that the random variables are independent and identically distributed. Also, that the law is true only in the limit. If we have a finite number of random variables, the sample mean will be close to the distribution mean, but not necessarily equal.

Also it is important to mention that the law refers to the average of the sample mean, that is, $\sum_{i=1}^n \frac{X_i}{n}$ and it is not necessarily true for other formulas, like for example, the deviation from the theoretical values $\sum_{i=1}^n X_i - n \times \bar{X}$ which not only it does not converge, but that it increases in absolute value as n increases (see Example 3.13).

■ **Example 3.13** If we toss a fair coin, the probability that the outcome will be head is equal to 1/2. According to the law of large numbers, the proportion of heads in a large number of coin tosses will be close to 1/2. However, the difference between the number of heads and tails will not be close to zero. In fact, the larger the number of coin tosses, the larger will be this difference. This is a highly counterintuitive fact, since most of the people think that the more we toss the coin, the closer will be the number of heads to the number of tails, which is not true. ■

3.7.2 Central Limit Theorem

Let X_1, \dots, X_n be a sample of n independent and identically distributed random variables with mean μ and variance σ^2 . As we saw in the previous section, the law of large numbers states that the sample average \bar{X}_n converges in probability to μ as n increases. The central limit theorem states that the distribution of the difference between the sample average \bar{X}_n and the population mean μ , when multiplied by the factor \sqrt{n} approximates to the normal distribution with mean 0 and variance σ^2/n . The theorem is true regardless of the shape of the original random variables.

Theorem 3.7.4 — Central Limit Theorem. Let X_1, \dots, X_n be a random sample of size n from a distribution with mean μ and a finite variance σ^2 . Then for each fixed number $\varepsilon > 0$ we have that

$$\lim_{n \rightarrow \infty} Pr\left(\frac{\bar{X}_n - \mu}{\sigma/\sqrt{n}} \leq \varepsilon\right) = \Phi(x)$$

where $\Phi(x)$ denotes the cumulative distribution function of the standard normal distribution.

Proof. Perhaps is too complex for this book to give a proof of the theorem. ■

TODO: Use a formulation of the theorem that does not uses the cumulative distribution

The Central Limit Theorem is a fundamental concept in probability theory used in statistical analysis and inference. It allows us to compute the probability that the sample average is close to the distribution mean. It is important to recall the conditions for the central limit theorem to be true: the samples must be independent and identically distributed, the original distribution has to have a finite variance, and the sample size must be sufficiently large.

■ **Example 3.14** Start with a uniform distribution, for example, throwing a dice. Explain and show how the arithmetic mean can be described with a binomial distribution. Explain that going to the limit results in the normal distribution. ■

Elaborate in the fact the central limit theorem, and the law of large numbers, do not help too much if our target metric is not the arithmetic mean. Mention the implications, for example, in machine learning.

References



4. Computability

*Caminante, no hay camino,
se hace camino al andar.¹*

Antonio Machado

We begin our review of the background required to understand the theory of nescience by providing a mathematical formalization of the concept of a *computable procedure*. Intuitively, a computable procedure is a method consisting of a finite number of instructions that, when applied to a problem, produce the correct answer after a finite number of steps. The key point is that the instructions must be clear and precise enough for any human to follow without aid. We can even go a step further and require that the instructions must be so straightforward that a machine could execute them. In 1936, British mathematician Alan Turing introduced a formal model for a family of hypothetical machines and posited that for every computable procedure (in its intuitive sense), there exists a *Turing machine* capable of computing it. The model was not only simple enough for precise mathematical analysis but also versatile and powerful.

Over the years, many alternative proposals have aimed to formalize the concept of computable procedure. Some have been very complicated, but all have proven equivalent to the concept of the Turing machine; that is, they solve the same set of problems. Two notable examples of alternative definitions are the *lambda calculus* by Alonzo Church and the *theory of recursive functions* by Kurt Gödel and Stephen Kleene. The *Church-Turing thesis* asserts that any formalization of the concept of a computable procedure, meeting some minimum requirements (such as performing a finite amount of work in a single step), is equivalent to a Turing machine. This thesis suggests an objective notion of a computable procedure that is independent of any specific formalization.

The concept of the Turing machine, initially referring to mechanical devices designed to solve individual problems, has been extended and universalized. A *universal Turing machine* exists that can resolve all computable problems by simulating the behavior of other specific machines, akin to how modern computers run algorithms written in various programming languages. This concept raises a significant question: Are there problems that are not computable? We will see

¹Wanderer, there is no road, the road is made by walking.

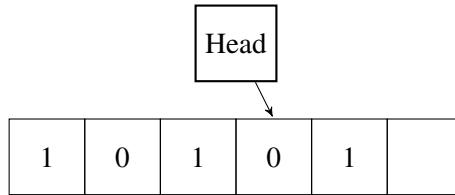


Figure 4.1: Turing Machine

that the answer is affirmative, certain well-defined problems exceed the computational capabilities of computers, and such problems are more common than initially anticipated. The notion of uncomputable functions will be pivotal in our theory of nescience.

Given the abstract nature of most entities studied in science, we employ the concept of the *oracle Turing machine* to aptly formalize our theory. An oracle Turing machine resembles a regular Turing machine but is augmented with the capability to query an external oracle, whose workings are not fully understood, to aid in its computations. This oracle can address problems that are unresolvable by standard Turing machines—essentially, it can solve uncomputable problems. The oracle is a theoretical construct that represents a source of solutions or information that is not bound by the limitations of computability. It acts as a ‘black box’ that instantly provides answers to specific questions or problems, enabling the oracle Turing machine to transcend its computational boundaries. The oracle is an abstract and non-mechanical entity, a theoretical tool used to explore the bounds of computation, rather than a physical or concrete machine that can be built or observed.

Turing machines illuminate the inherent limitations of our computational capabilities. This exploration into the abstract and theoretical realms of computation is not just a philosophical endeavor; it also possesses practical applications in the field of *computational complexity*. Located at the intersection of computer science and mathematics, computational complexity evaluates the challenges associated with solving problems, measured against the required resources, notably time and space. Problems are classified based on their intrinsic complexity and the computational effort required for their resolution. One of the key questions in this field is the elusive and yet unsolved $P \stackrel{?}{=} NP$ question, which seeks to determine if the class P of problems, those that are easy to solve, coincides with the class NP of problems, whose solutions are easy to verify. In this book, our focus is not solely on the epistemological question of identifying which problems can be effectively solved given ample time and space, but also on those that can be resolved efficiently in time.

4.1 Turing Machines

A Turing machine is an extremely simplified model of a general-purpose computer, yet it is capable of solving any problem that real computers can address. Intuitively, one can envision the machine as consisting of a head that operates on a two-way infinite tape striped with symbols. At each time step, the machine reads the symbol under the head and decides to either write a new symbol on the tape, move the head one square to the left or right, or execute both actions. Algorithms are implemented using an internal table of rules housed within the control head, and the actual input to the algorithm is encoded on the tape. Once the machine reaches its final state, the algorithm’s output can be read from the tape. Figure 4.1 depicts an example of a machine in its initial state, with the head located at the beginning of the input string.

The following definition formally introduces the concept of a Turing machine.

Definition 4.1.1 — Turing Machine. A *Turing machine* is a 7-tuple $(Q, \Gamma, \sqcup, \Sigma, q_i, q_f, \tau)$ where:

- Q is a finite, non-empty, set of *states*,
- Γ is a finite, non-empty, set of *tape symbols*,
- $\sqcup \in \Gamma$ is the *blank symbol*,
- $\Sigma \subseteq \Gamma \setminus \sqcup$ is the set of *input symbols*,
- $q_o \in Q$ is the *initial state*,
- $q_f \in Q \setminus \{q_o\}$ is the *final state*,
- $\tau : (Q \setminus \{q_f\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$ is a partial *transition function*.

The algorithm executed by the machine is defined by the transition function τ . This function dictates the machine's actions based on its current state and the tape symbol currently under the head. According to τ , the machine transitions to a new state, writes a new symbol on the tape (or retains the existing one), and moves the head left, right, or keeps it stationary (L , R , or S respectively). The machine follows a finite, uniquely determined sequence of steps until it reaches the final state q_f and *halts*, making no subsequent moves. The algorithm's output is the string of symbols $s \in \Sigma^*$ remaining on the tape after halting. Some machines, however, may enter an infinite loop, never reaching a halting state. If a machine encounters an undefined transition, it will enter in an infinite loop, never halting.

The machine's input consists of a string of symbols, with the assumption that the machine's head is initially positioned at the first symbol of the input string. To address problems involving an object O that isn't a string, we must first develop a method to encode that object as a string, denoted as $\langle O \rangle$.

■ **Example 4.1** The following Turing machine is designed to solve the problem of adding two natural numbers. It consists of the set of states $Q = \{q_o, q_1, q_f\}$, the set of tape symbols $\Gamma = \{0, 1, \sqcup\}$, and the set of input symbols $\mathcal{B} = \{0, 1\}$. The transition function is defined in the table below, where rows are indexed by machine states, and columns by tape symbols:

	0	1	\sqcup
q_o	(q_f, \sqcup, S)	(q_1, \sqcup, R)	\uparrow
q_1	$(q_f, 1, S)$	$(q_f, 1, R)$	\uparrow

Table 4.1: Transition Rules

For natural numbers n and m , the input string is composed of n occurrences of the symbol '1', followed by a '0', and then followed by m occurrences of '1'. The machine's output will be a string of $n + m$ consecutive '1's. For instance, to add the numbers 2 and 3, the input string should be $\sqcup 110111 \sqcup$, resulting in the output string $\sqcup 11111 \sqcup$. ■

A Turing machine can also be represented by a *state diagram*. A state diagram is similar to a labeled directed graph² where the vertices represent the states of the machine. The edges signify transitions from one state to another, and the edge labels indicate the symbol under the head that leads to the new state, the symbol that gets written on the tape, and the direction in which the head moves. Following these conventions, the state diagram for the Turing machine in Example 4.1 is depicted in Figure 4.2.

It is a remarkable fact that minor alterations to the definition of a Turing machine do not change its computational power. In other words, the definition is highly robust. In Example 4.2, it's demonstrated that adding more tapes to the machine doesn't expand the range of problems it can

²In this particular case, we allow loops and multiple edges originating from vertices.

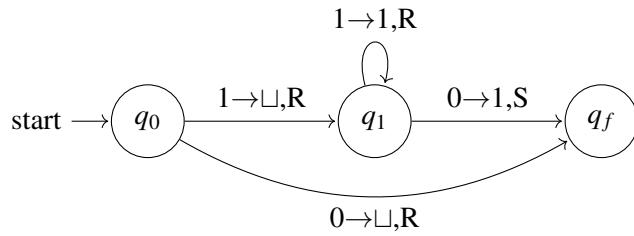


Figure 4.2: Example of Turing Machine

solve. Similar arguments can be made when adding finite storage to the control tape, allowing for parallel processing with multiple control heads, and so on.

■ **Example 4.2** A *multitape Turing machine* is a Turing machine equipped with multiple heads and their respective tapes. In the initial configuration, the input string resides in tape 1, while the other tapes are blank. The transition function for a multitape Turing machine is:

$$\tau : (Q \setminus q_f) \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R, S\}^k,$$

where k denotes the number of tapes. Multitape Turing machines are equivalent in power to standard Turing machines. We can validate this claim by devising a method for a standard Turing machine to mimic a multitape machine's behavior. This requires encoding the content of multiple tapes onto a single tape, introducing a new symbol as a tape separator, and encoding the positions of the heads across the tapes with a distinct head location symbol. If we designate the tape separation symbol as $|$ and the head location symbol as h , a simulation tape for a machine with 3 tapes might appear as $\sqcup 01h00|000h1|h0101\sqcup$. The standard machine's operation would involve scanning the subtapes one by one, pinpointing the head's location, and executing the necessary transition. If the computation on one subtape necessitates writing a new symbol beyond its boundary, we'd need to shift subsequent symbols to accommodate the new one. While the simulation might operate at a slower pace than the original multitape machine, both machine types can solve an identical set of problems. ■

For the remainder of this book, without any loss of generality, we'll assume that the set of input symbols is $\Sigma = \mathcal{B}$ and the set of tape symbols is $\Gamma = \{0, 1, \sqcup\}$.

In addition to providing a formal definition of a Turing machine, it's essential to formally outline its computational process. This entails detailing how the machine reads the input string, produces the output string, and transitions between states during computation. We will start by defining the concept of the machine's internal configuration. This configuration captures the machine's current state and position, as well as the present state of the tape.

Definition 4.1.2 A *configuration* of a Turing machine T is the 3-tuple (q, s, i) , where $q \in Q$ represents a state of the machine, $s \in \Gamma^+$ denotes a string containing the tape's content (excluding the blank symbols), and $1 \leq i \leq n$ is the index of the symbol s_i beneath the head. Here, s_1 is the first non-blank symbol on the tape, and $n = l(s)$.

Configurations enable us to describe the current state of a Turing machine without any loss of information. At any stage of computation, one could halt the machine, record its configuration, and later resume the computation from the exact point of interruption using this configuration.

The following definition explains how we transition from one configuration to the next during computation.

Definition 4.1.3 A configuration $C = (q, s, i)$ yields another configuration $C' = (r, s', j)$ if there exists a transition $\tau : (q, s_i) = (r, s'_i, a)$, where $s = s_1 \dots s_{i-1} s_i s_{i+1} \dots s_n$, $s' = s_1 \dots s_{i-1} s'_i s_{i+1} \dots s_n$, and

$$j = \begin{cases} i+1 & \text{if } a = R \\ i-1 & \text{if } a = L \\ i & \text{if } a = S \end{cases} \quad (4.1)$$

Building on the concepts of configuration and one configuration yielding another, we can now formally articulate the notion of computation.

Definition 4.1.4 — Computation. Let T be a Turing machine, C_0 its initial configuration, and C_n a configuration encompassing the final state q_f . A *computation* under machine T refers to a finite sequence of $n + 1$ configurations (C_0, C_1, \dots, C_n) wherein each configuration C_k yields the subsequent configuration C_{k+1} , for all $0 \leq k < n$.

Computations are deterministic; meaning, for a given Turing machine T and an input string s , the configuration sequence is preordained. If machine T neither halts nor progresses with input s , we deduce the absence of computation.

■ **Example 4.3** The computation of the Turing machine described in Example 4.1 using the input string 110111 results in the following sequence of configurations:

- 1 $(q_0, 110111, 1)$
- 2 $(q_1, 10111, 1)$
- 3 $(q_1, 10111, 2)$
- 4 $(q_f, 11111, 2)$

Intuitively, a procedure is deemed computable by a human if it can be delineated through specific steps, executed systematically, without relying on intuition or ingenuity. This intuitive grasp aligns with the formalized concept of a Turing machine, bridging informal comprehension and the machine's rigorous definition—a cornerstone in the theory of computation. However, this alignment presents an intriguing challenge. Affirming that our grasp of computability mirrors a Turing machine's capabilities cannot be proven traditionally, as 'computability' lacks a well-defined interpretation. Consequently, some researchers categorize this as a *thesis*, avoiding the formal 'theorem' label. Turing himself opted to term it a *definition*, steering clear of denoting it as a theorem.

Theorem 4.1.1 — Turing's Thesis. A procedure is computable if, and only if, it can be executed by a Turing machine.

To further underscore the significance and robustness of the Turing machine as a model of computation, it's worth noting, as mentioned earlier in this chapter, that all alternative formalizations of computability proposed to date align in terms of their computational capabilities with that of the Turing machine. This universality underscores the Turing machine's central position in the realm of theoretical computer science.

4.2 Universal Turing Machines

In Section 4.1, we explored storing the current state of a Turing machine, its configuration, to pause and later resume computation. In Example 4.4, we will delve into a similar procedure, not for storing the machine's current state, but for saving a comprehensive description of the machine itself. This methodology facilitates the enumeration, or listing, of all possible Turing machines. Such

enumeration is instrumental in demonstrating the existence of problems that cannot be solved by any Turing machine (refer to Section 4.3) and unveiling the pivotal concept of the *Universal Turing Machine*.

■ Example 4.4 To describe a Turing machine concisely, we need to encode the transition function $\tau : (Q \setminus \{q_f\}) \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$. This function can be represented as a collection of quintuples (q, s, r, t, a) , where $q \in (Q \setminus \{q_f\})$, $r \in Q$, $s, t \in \Gamma$, and $a \in \{L, R, S\}$. In this manner, any Turing machine T is fully described by a collection of quintuples:

$$(q_1, s_1, r_1, t_1, a_1), (q_2, s_2, r_2, t_2, a_2), \dots, (q_m, s_m, r_m, t_m, a_m)$$

where $m \leq d(Q \setminus \{q_f\}) \times d(\Gamma)$, with the stipulation that the first quintuple refers to the initial state and the second one to the final state; i.e., $q_1 = q_o$ and $r_2 = q_f$. A possible approach to describe these quintuples is to encode the elements of the set $Q \cup \Gamma \cup \{L, R, S\}$ using a fixed-length binary code (refer to Definition 5.1.6 for more details), encoding the quintuple (q, s, r, t, a) as $\langle q, s, r, t, a \rangle$. The length of an encoded quintuple is $5l$, where $l = \lceil \log(d(Q \cup \Gamma \cup \{L, R, S\})) \rceil$. Following this convention, machine T is encoded as the binary string:

$$\langle T \rangle = \langle \bar{l}, \langle q_1, r_1, s_1, t_1, a_1 \rangle, \dots, \langle q_r, r_r, s_r, t_r, a_r \rangle \rangle$$

The length of the encoded machine, following this schema, would be $l(\langle T \rangle) \leq 5lm + \log l + 1$. ■

Since each Turing machine is composed by a finite set of quintuples, we can encode and list all the machines using a shortlex ordering. We associate each machine T with the index i corresponding to its position in this list, and we denote by T_i the i -th Turing machine. Each positive integer i encodes one, and only one, Turing machine. However, as Proposition 4.2.1 shows, all Turing machines have an infinite number of indexes. We associate each Turing machine with its smallest index.

Proposition 4.2.1 — Padding Lemma. Each Turing machine has infinitely many indexes.

Proof. Consider a Turing machine T_i encoded by the string $\langle T_i \rangle$. We can create a new encoding $\langle T_j \rangle$ by appending a finite number of 0's to $\langle T_i \rangle$, such that $\langle T_j \rangle = \langle T_i \rangle 0^n$ for some positive integer n . Since n can take on any positive integer value, there are infinitely many possible encodings $\langle T_j \rangle$ for the same Turing machine T_i . ■

A universal Turing machine is a machine that can simulate the behavior of any other Turing machine on arbitrary input. The universal machine achieves this by reading both the description of the machine to be simulated (for instance, using the coding schema described in Example 4.4) and the input string for the computation from its own tape.

■ Definition 4.2.1 — Universal Turing Machine. A *Universal Turing Machine* is a Turing machine U such that $U(\langle \langle T_i \rangle, s \rangle) = T_i(s)$ for all Turing machines T_i and all input strings $s \in \mathcal{B}$.

Naturally, we must prove that such a machine exists before we can utilize it. One could argue that a human being could decode the machine T_i and simulate its behavior with the input string s , and then refer to Theorem 4.1.1. A more rigorous approach would be to explicitly construct a universal Turing machine. However, providing a detailed description of one of these machines is beyond the scope of this book. Instead, we direct the reader to the references included at the end of the chapter for further exploration.

4.3 Non-Computable Problems

Turing machines enable us to delineate the set of problems that can be resolved through effective procedures or, in other words, by computers. It may be surprising to learn that numerous problems

cannot be addressed using algorithms; such challenges lie beyond the computational capabilities of machines. We are not alluding to speculative queries like whether a computer can be intelligent or self-aware but to concrete, well-defined mathematical problems. We are also not referring to complex problems that demand a substantial amount of time to solve, as those, irrespective of their time consumption, remain computable.

One classic exemplar of non-computability is the *halting problem*. As illustrated in Algorithm 4.1, it involves a program or algorithm tasked with determining whether any given program (including itself) and input will eventually halt or continue to run indefinitely. Alan Turing proved that no algorithm can exist to solve this problem for all possible program-input pairs. This revelation wasn't a reflection on the limitations of technology or processing power but highlighted a profound theoretical limit intrinsic to computation.

Algorithm 4.1 HALT function

```

procedure HALT( $A, I$ )
  if  $A(I)$  halts then
    return 1
  else
    return 0
  end if
end procedure
```

The proposition below proves that the halting problem is non-computable.

Proposition 4.3.1 — Halting Problem. Define HALT as in Algorithm 4.1. There does not exist a Turing machine that computes the HALT function for all possible pairs (A, I) , where A is a Turing machine and I is the input string to that machine.

Proof. The proof is by contradiction. Assume that the machine $HALT$ exists, and define a new Turing machine TC such that $TC(A) = 1$ if $HALT(A, A) = 0$, and $TC(A)$ will never stop if $HALT(A, A) = 1$. Then the contradiction arises when we ask about the result of $TC(TC)$: if $TC(TC)$ stops we have that $HALT(TC, TC) = 0$ and that $TC(TC)$ should not stop, and if $TC(TC)$ does not stop then we have that $H(TC, TC) = 1$ and thus $TC(TC)$ should stop. ■

The existence of such non-computable problems underscores the boundaries of mechanical computation. It illustrates that while Turing machines, and by extension, computers are profoundly powerful tools capable of solving an extensive array of problems, they are not omnipotent. A frontier of unsolvable problems exists, necessitating deeper exploration into the realms of mathematics, logic, and perhaps even philosophy to understand the inherent limits of computation.

The Halting Problem also has significant practical consequences in computer programming. For instance, it is impossible to write a program that can guarantee any other arbitrary program is bug-free or that all infinite loops with conditional exits will eventually halt for all possible inputs.

The next example introduces a well-defined, practical problem involving simple string manipulation that cannot be solved using computers.

■ **Example 4.5** Given two finite lists $(\alpha_1, \dots, \alpha_n)$ and $(\beta_1, \dots, \beta_n)$ of strings over some alphabet Σ , where $d(\Sigma) \geq 2$, the *Post Correspondence Problem* (PCP) asks to determine if there exists a sequence of $K \geq 1$ indices (i_k) , with $1 \leq i_k \leq n$ for all $1 \leq k \leq K$, such that $\alpha_{i_1} \dots \alpha_{i_K} = \beta_{i_1} \dots \beta_{i_K}$. For instance, given the sequences (a, ab, bba) and (baa, aa, bb) , a solution would be $\alpha_3 \alpha_2 \alpha_3 \alpha_1 = \beta_3 \beta_2 \beta_3 \beta_1$. No algorithm exists to solve PCP. Like many proofs of incomputability, the proof proceeds by showing that HALT can be reduced to PCP, meaning if PCP is decidable, then the Halting Problem should be decidable as well. We will not detail the proof in this section; for interested readers, we refer to the references at the end of this chapter. ■

Non-computable problems are generally not derived directly from natural phenomena but from logical and mathematical constructs. To date, there are no known examples of non-computable problems manifesting plainly in natural phenomena. It's essential to distinguish between non-computability and unpredictability. Non-computable problems are those for which no algorithm can ever be created to solve them. In contrast, unpredictable systems (such as chaotic or complex systems) are theoretically computable but are unpredictable in practice due to factors like sensitivity to initial conditions or measurement precision.

4.4 Computable Functions and Sets

Each Turing machine T defines a function $f_T : \mathcal{B}^* \rightarrow \mathcal{B}^*$ that assigns to each input string $s \in \mathcal{B}^*$ an output string $T(s) \in \mathcal{B}^*$. The relationship between Turing machines and functions forms the basis for introducing the concept of a *computable function*.

Definition 4.4.1 A function $f : \mathcal{B}^* \rightarrow \mathcal{B}^*$ is *computable* if there exists a Turing machine T that defines the function f and halts for all the values of \mathcal{B}^* .

The terminology in computational theory can vary. While computable functions are occasionally referred to as *recursive functions*, this book opts for the term computable functions for consistency.

■ **Example 4.6** The function that assigns to each pair of natural numbers x and y their sum $x + y$ is computable, as demonstrated in Example 4.1. ■

In real-world scenarios, certain functions don't provide a defined output for all possible inputs. Partial computable functions, characterized by Turing machines that don't halt for specific inputs, model these cases.

Definition 4.4.2 A partial function $f : \mathcal{B}^* \rightarrow \mathcal{B}^*$ is *partial computable* if there exists a Turing machine T that defines f for defined values and does not halt for undefined values.

The distinction between total computable functions and partial computable functions is significant in computability theory because it reflects the difference between problems that are always solvable by an algorithm (total) and those that are only solvable in some cases (partial).

■ **Example 4.7** The function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ that assigns to each pair of natural numbers x and y the number $x - y$ is a partial computable function, since it is not defined in the case that $x < y$. Recall that according to our definition of Turing machine (see Definition 4.1.1), when the machine reaches an undefined configuration enters an infinite loop without ever halting. ■

We can expand the application of the principles of computability and partial computability to the domain of sets. We characterize sets through the lens of their characteristic functions that discern the membership of elements within the sets.

Definition 4.4.3 A set $A \in \mathcal{B}^*$ is *computable* if its characteristic function χ_A is a total computable function. A set $A \in \mathcal{B}^*$ is *computably enumerable* if its characteristic function χ_A is a partial computable function, that is, $\chi_A(a) = 1$ if $a \in A$, but $\chi_A(a)$ is undefined if $a \notin A$.

The application of these concepts is illustrated through the example of the set of all Turing machines that halt for all inputs.

■ **Example 4.8** The set of all Turing machines that halt on all inputs, as demonstrated in 4.3.1, is not computable but is computably enumerable. ■

Next proposition provides an alternative characterization of computable sets.

Proposition 4.4.1 A set $A \in \mathcal{B}^*$ is computable if and only if A and its complement A^c are computably enumerable.

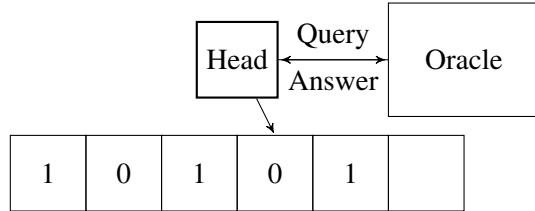


Figure 4.3: Oracle Turing Machine

Proof. If A is computable, by definition, there exists a Turing machine that decides for any input $x \in \mathcal{B}^*$ whether $x \in A$ or $x \notin A$, halting in both cases. This implies that both A and its complement A^c can be enumerated by Turing machines. Thus, both A and A^c are computably enumerable.

Conversely, suppose A and A^c are computably enumerable. This means there exist two Turing machines, T_A and T_{A^c} , that enumerate the elements of A and A^c , respectively. To show that A is computable, construct a Turing machine T that, given an input $x \in \mathcal{B}^*$, simulates T_A and T_{A^c} in parallel to search for x . If x appears in the enumeration produced by T_A , T halts and accepts x as an element of A . If x appears in the enumeration produced by T_{A^c} , T halts and accepts x , indicating $x \notin A$. Since every element of \mathcal{B}^* must be in either A or A^c and both sets are computably enumerable, T will eventually halt for every input x , proving that A is computable. ■

4.5 Oracle Turing Machine

An oracle Turing machine (see Figure 4.3) is a theoretical model of computation that extends the capabilities of a standard Turing machine by providing it with an oracle. The oracle is a black box that can instantly compute certain answers, even for problems that are unsolvable or would take an impractical amount of time for a standard Turing machine to process. This model helps computer scientists and mathematicians explore the implications and boundaries of computational theory, including questions about complexity classes and the limits of what is computationally possible. The oracle Turing machine isn't a physical or implementable machine but rather a conceptual tool used in theoretical studies.

Definition 4.5.1 — Oracle Turing Machine. An *oracle Turing machine* with oracle set \mathcal{O} is a 8-tuple $(Q, \Gamma, \sqcup, \Sigma, q_i, q_f, \tau, \mathcal{O})$ where:

- Q is a finite, non-empty, set of *states*,
- Γ is a finite, non-empty, set of *tape symbols*,
- $\sqcup \in \Gamma$ is the *blank symbol*,
- $\Sigma \subseteq \Gamma \setminus \sqcup$ is the set of *input symbols*,
- $q_o \in Q$ is the *initial state*,
- $q_f \in Q \setminus \{q_o\}$ is the *final state*,
- $\tau : (Q \setminus \{q_f\}) \times \Gamma \times \{0, 1\} \rightarrow Q \times \Gamma \times \{L, R, S\}$ is the *transition function*,
- $\mathcal{O} \subseteq \Sigma^*$ is the *oracle set*.

Building on the concept of a regular Turing machine (refer to Definition 4.1.1), an oracle Turing machine introduces the unique feature of an oracle set \mathcal{O} . This set comprises a subset of strings for which the oracle can instantly provide answers. The true strength of an oracle machine emerges when the set \mathcal{O} is non-computable; in cases where \mathcal{O} is computable, a regular Turing machine would be sufficient.

The transition function τ for the oracle Turing machine is nuanced. At each step the machine

will query the oracle. Specifically, it sends the current string w —starting under the head and extending to the rightmost non-empty cell of the input tape—to the oracle. The oracle then responds with a binary answer: '1' if w is in \mathcal{O} or '0' if it isn't. The machine doesn't always utilize this response. There are steps where, despite receiving an answer, the computation proceeds unaffected by the oracle's response—essentially ignoring it. However, when the oracle's answer is pivotal, the machine makes a decision based on it. This decision could affect the next state, the next symbol to be written, or the next move (left, right, or stay). In essence, while the machine has the capability to continuously consult the oracle, it strategically chooses when to act on the information received. We could have modified the τ function in such a way that the oracle is queried only when the answer is relevant for the computation, but that would require making important changes to the behaviour of the function, such as the introduction of new control states.

■ **Example 4.9** In the realm of theoretical computer science, an oracle can be invoked to "solve" the halting problem (refer to Theorem 4.3.1). The oracle is a hypothetical device or black box that, as if by magic, provides an instantaneous answer to a specific problem instance. The oracle set \mathcal{O} would consist of a collection of strings in the form $\langle P, I \rangle$, where P represents a program encoded as a string and I denotes its input. Given a program and its input, this oracle would instantly inform us whether the program halts on that input. Naturally, this concept is purely theoretical. No such oracle exists in reality, and the halting problem remains unsolvable in practical terms. ■

Turing machines are a subset of oracle Turing machines

Proposition 4.5.1

The machine is independent of the oracle set

Proposition 4.5.2

An oracle machine is characterized by two independent components: a particular Turing machine and a particular oracle set.

How to encode oracle Turing machines

■ Example 4.10

What means a function or set to be oracle computable

■ Definition 4.5.2

4.6 Computational Complexity

Computational complexity refers to the study of the efficiency of algorithms in terms of the resources they consume, such as time and space, to solve a given problem. By categorizing problems based on their inherent difficulty, computational complexity provides insights into the feasibility of solving problems within practical limits. It involves classifying problems into *complexity classes* offering a framework to analyze and compare the performance of algorithms. In this book we are interested mostly in the time required to solve a problem. We compute the running time of an algorithm as a function of the length of the string representing the input.

■ **Definition 4.6.1** Let M a Turing machine. The running time or *time complexity* of M is the function $T_M : \mathbb{N} \rightarrow \mathbb{N}$, where $T_M(n)$ is the maximum number of steps that machine M takes for any input of length n .

Big-O notation is a mathematical notation that describes the upper bound of an algorithm's running time in the worst-case scenario, serving as a measure of its efficiency. Big-O notation provides a high-level analysis of an algorithm's performance, offering insights into how the running time requirements grow as the size of the input increases. It abstracts the details of the machine

model and focuses on the most significant factors that contribute to the growth rate of the algorithm's complexity.

Definition 4.6.2 Let $f : \mathbb{N} \rightarrow \mathbb{N}$ and $g : \mathbb{N} \rightarrow \mathbb{N}$ be two functions. We say that the function f is of order g , denoted $f(n) = O(g(n))$, if there exist constants $c > 0$ and $n_0 \geq 0$ such that for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$.

Big-O notation is instrumental in comparing different algorithms and selecting the most appropriate one for a given problem and dataset size.

■ **Example 4.11** Consider a polynomial function that represents the time complexity of an algorithm:

$$f(n) = 3n^3 + 2n^2 + 5n + 7$$

In the context of big-O notation, we are interested in the term with the highest growth rate as n approaches infinity, because it will eventually dominate the function. In this case, the term $3n^3$ has the highest growth rate. Thus, we have

$$f(n) = O(n^3)$$

■

A complexity class consists of a collection of problems that are classified together due to their shared level of computational complexity. Each problem within a class shares similar characteristics of time complexity, ensuring a consistent measure of computational effort needed for resolution. These classes offer valuable insights and are integral in assessing and distinguishing the practical solvability and resource requirements of various computational challenges.

Definition 4.6.3 A complexity class C is a collection of problems for which there exists a time bound function $f(n)$ such that any problem P in C can be solved by a Turing machine M with time complexity $T_M = f(n)$, where n is the size of the input.

The class P consists of problems that can be solved in polynomial time by a Turing machine. In other words, for every problem in P , there exists an algorithm that can determine the solution in a number of steps that is a polynomial function of the size of the input. This class is foundational in computational complexity, serving as a baseline for measuring the efficiency of algorithms.

Definition 4.6.4 The class problems that can be solved in polynomial time, denoted by P , is defined as the collection of problems for which there exists a Turing machine M that solve them and a polynomial $p(n)$ such that the time complexity of M is $p(n)$.

Problems within P are considered tractable, meaning they can be practically solved even for large inputs. The concept of polynomial time solvability is crucial in distinguishing between problems that have efficient solutions in practice and those that do not.

■ **Example 4.12** A classic example of a problem in class P is the *shortest path problem*. In this problem, you're given a weighted graph and two vertices, and the objective is to find the path of minimum total weight between these two vertices. One common algorithm to solve this problem is Dijkstra's algorithm. It works by iteratively selecting the vertex with the smallest known distance from the start vertex, and updating the estimated distances to its neighbors. This algorithm runs in polynomial time, specifically $O(V^2)$ for a graph with V vertices. Since there exists a polynomial-time solution to this problem it is in class P .

■

A *verifier* is a Turing machine that, given an input string and an additional string known as a *certificate* or witness, determines whether the input satisfies a particular property, resulting in a

binary "yes" or "no" answer. The core idea behind a verifier is not necessarily to find a solution to a problem, but rather to efficiently check or "verify" the validity of a proposed solution.

Definition 4.6.5 A verifier is a Turing machine M that, given an input string x and an additional certificate string y , determines if x belongs to a certain subset S of strings. The machine satisfies two conditions:

- 1 If x belongs to subset S , there exists a certificate y such that $V(x, y) = 1$.
- 2 If x does not belong to subset S , then for every possible certificate y , $V(x, y) = 0$.

Being the verifier a Turing machine, we can measure its time complexity, in terms of the length of the certificate. When a problem has solutions that can be verified in polynomial time using a verifier, it sheds light on the inherent complexity of that problem, situating it within specific computational classes.

Definition 4.6.6 A problem is in the class NP if there exists a polynomial $p(n)$ and a Turing verifier V such that for every instance x :

- 1 If x has a solution in subset S , then there exists a certificate y with length at most $p(|x|)$ such that V confirms x as a valid instance of S using y in polynomial time.
- 2 If x does not have a solution in subset S , then for every possible certificate y with length at most $p(|x|)$, V does not confirm x as a valid instance of S .

■ **Example 4.13 TODO: Rewrite** A classic example of a problem in class NP is the *Traveling Salesman Problem* or TSP: given a list of cities and the distances between each pair of cities, the problem is to find the shortest possible route that visits each city exactly once and returns to the original city. To show that TSP is in NP , we don't need to demonstrate how to efficiently find the shortest route; we only need to show that, given a proposed route (or tour), we can efficiently verify whether that route satisfies the conditions of the problem and is shorter than or equal to a certain length. The verification process would be:

- Check if the proposed route visits each city exactly once and returns to the starting city. This can be done by traversing the proposed route and marking visited cities.
- Compute the total distance of the proposed route by summing up the distances between consecutive cities in the route.
- Compare the computed distance to the given threshold or limit.

Given a proposed solution (the route) and the distances between cities, a verifier can perform the above steps in polynomial time with respect to the number of cities, thereby verifying the solution's validity. In the context of the NP definition, the "certificate" for TSP would be the proposed route. If TSP has a route shorter than or equal to a specific distance, then there exists a certificate (the route itself) that can be verified in polynomial time. If not, no such certificate would make the verifier confirm the solution. ■

The $P = NP$ problem is one of the most fundamental questions in computer science. It asks whether every problem for which a potential solution can be verified quickly can also have its solution found quickly. In essence, if P equals NP , it would mean that finding solutions is as "easy" as checking them. Despite extensive study, no one has been able to prove that $P = NP$ or $P \neq NP$, making it a longstanding open challenge in theoretical computer science.

Theorem 4.6.1 — P=NP Problem. The class of problems P is equal to the class of problems NP .

If P were to equal NP , it would signify a monumental shift in our understanding of computational tasks. Practically, many problems considered intractable today, especially those in cryptography, would become tractable. Secure communication over the internet relies on certain problems being hard to solve (like factoring large numbers). If $P = NP$, then encryption methods

underpinning online security could be broken, potentially leading to a crisis in digital security. Conversely, numerous challenges in fields like optimization, drug discovery, and logistics could see groundbreaking solutions, revolutionizing industries. While it's tempting to view $P = NP$ solely in the context of its challenges, especially to security, it would also herald unprecedented opportunities and advancements in multiple domains.

Turing reducibility is a fundamental concept in the realm of theoretical computer science and computability theory. It provides a means to compare the "computational difficulty" of problems. Specifically, a problem A is Turing reducible to a problem B if there exists an algorithm for solving A that may make use of a subroutine solving B an arbitrary number of times.

Definition 4.6.7 Let A and B be two problems. We say that A is **Turing reducible** to B , denoted by $A \leq_T B$, if there exists a Turing machine that solves A by making use of one or more Turing machines that solve B .

If A is Turing reducible to B and vice versa, then the two problems are said to be Turing equivalent, indicating they have the same computational complexity.

■ **Example 4.14** *TODO: rewrite.* Consider the problems of GRAPH ISOMORPHISM and SUBGRAPH ISOMORPHISM.

GRAPH ISOMORPHISM (GI): Given two graphs G_1 and G_2 , is there a one-to-one correspondence (isomorphism) between their vertices such that the edges are preserved?

SUBGRAPH ISOMORPHISM (SI): Given two graphs G_1 and G_2 , does G_1 contain a subgraph that is isomorphic to G_2 ?

The GRAPH ISOMORPHISM problem is Turing reducible to the SUBGRAPH ISOMORPHISM problem. Here's the reasoning:

If we have an efficient solver for SUBGRAPH ISOMORPHISM, then we can use it to solve GRAPH ISOMORPHISM. For two graphs G_1 and G_2 of the same size, we ask the SI solver if G_2 is a subgraph of G_1 and vice versa. If both answers are affirmative, then G_1 and G_2 are isomorphic. Thus, an efficient solution for SI can be used to determine a solution for GI.

This example demonstrates the essence of Turing reducibility: by solving the more general problem (SUBGRAPH ISOMORPHISM), we inherently find a solution for the more specific problem (GRAPH ISOMORPHISM). ■

NP-hard

NP-Complete

References

TODO: Briefly mention the historical emergence of the concept, including the works of pioneers like Alan Turing, Alonzo Church, and others.

The original paper from Alan Turing where the concepts of Turing machine, universal Turing machine, and non-computable problems were introduced is [[turing1936computable](#)], however it is a difficult to read paper for the contemporary reader. An easier to read introduction to computability theory, from the point of view of languages, can be found in [[sipser2012introduction](#)], and a more advanced introductions in [[cooper2003computability](#)] and [[soare2016turing](#)]. In [[fernandez2009models](#)] we can find a description of the most important computability models proposed so far. The Post Correspondence Problem was introduced by Emil Post in [[post1946variant](#)]; for the details of the proof sketched in Example 4.5 please refer to [[sipser2012introduction](#)].

TODO: Here are a few seminal academic works on non-computability:

Turing, A. M. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. Proceedings of the London Mathematical Society, s2-42(1), 230-265. Content: This foundational paper by Alan Turing introduces the concept of Turing machines, laying the groundwork for

the theory of computation and establishing the halting problem's non-computability.

Church, A. (1936). An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, 58(2), 345-363. Content: Alonzo Church presents the λ -calculus and establishes Church's Thesis, claiming that his formalism captures the intuitive notion of "computable," and proving the unsolvability of the Entscheidungsproblem.

Gödel, K. (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme, I. *Monatshefte für Mathematik*, 38(1), 173-198. Content: Kurt Gödel's landmark paper where he introduces his incompleteness theorems, showing that within any sufficiently powerful mathematical system, there are statements that cannot be proven or disproven.

Post, E. (1946). A Variant of a Recursively Unsolvable Problem. *Bulletin of the American Mathematical Society*, 52(4), 264-268. Content: Emil Post presents a simpler, more accessible proof of unsolvability (non-computability) related to the halting problem and delves into the Post Correspondence Problem, a classic example of a non-computable problem.

Davis, M. (1958). *Computability and Unsolvability*. McGraw-Hill. Content: In this book, Martin Davis offers a comprehensive overview of the theory of computability and unsolvability, addressing topics like recursive functions, Turing machines, and Gödel's incompleteness theorems in an accessible manner.

These references should provide a solid academic foundation for exploring the multifaceted and intriguing world of non-computability. Each offers unique insights and perspectives that collectively illuminate the complexity and depth of this area of study.

TODO: Add a reference to how to build a universal Turing machine.

TODO: Introduce other well-known non-computable problems besides the halting problem, like Turing's "Entscheidungsproblem" or the Busy Beaver function.

TODO: Relate non-computability to Gödel's incompleteness theorems to illustrate the inherent limitations in formal mathematical systems.

TODO: Explore philosophical discussions on the implications of non-computability for artificial intelligence and human cognition.



5. Coding

Information is the resolution of uncertainty.

Claude Shannon

In this section, we are going to review the conceptual ideas and main results behind coding theory and the related area of information theory.

Coding is the process of describing a sequence of symbols from some alphabet by a sequence of symbols from another alphabet. Coding has many practical applications, such as error detection, cryptography, and telecommunications. Here, our interest is in data compression, that is, encoding a message using fewer symbols than its original representation, without losing any information. Compression algorithms reduce the size of messages by identifying unnecessary elements and removing them, usually by means of computing and eliminating statistical redundancies. For example, data compression can be achieved by assigning shorter descriptions to the most frequent symbols from the source, and longer descriptions to the less frequent symbols. A particular type of codes, the prefix-free codes, will be discussed as playing a central role in this book. Prefix-free codes allow us to link coding theory with probability theory, a link that will be very useful in the context of the theory of nescience.

Information theory proposes that the amount of information we receive when some event happens is the negative logarithm of the probability of that event. In this sense, the theory assumes that information is equivalent to surprise: the more unlikely an event is, the more information we receive when the event occurs. We are not going to use that interpretation of information in our theory of nescience, but we will extensively use another concept from information theory: entropy. Entropy quantifies the amount of uncertainty involved in the value of a random variable or the outcome of a random process. Entropy is important to us because it establishes a limit to the compression of texts: it is not possible to find a code with an average word length smaller than the entropy of the source alphabet.

There exist many interesting concepts derived from entropy, like joint entropy, conditional entropy, or mutual information. However, these concepts are more relevant in the context of communication because they allow us to solve the problem of how to transmit information in a

reliable manner over a noisy channel. Here, they are introduced for completeness purposes, and to compare them with our own definitions of joint nescience and conditional nescience.

5.1 Coding

Intuitively, coding refers to the process of losslessly describing a sequence of symbols (a message) coming from some alphabet by other sequences of symbols coming from a (potentially) different alphabet. There is no general agreement about what exactly a code is, as different authors propose different definitions in the literature. Fortunately, the definition of a prefix-free code, the kind of codes used in the theory of nescience, is a standard one.

Let $\mathcal{S} = \{s_1, s_2, \dots, s_q\}$ be a finite set called *source alphabet*, and $\mathcal{X} = \{x_1, x_2, \dots, x_r\}$ a finite set called *code alphabet*.

Definition 5.1.1 — Code. A *code* for \mathcal{S} is a total function $C : \mathcal{S} \rightarrow \mathcal{X}^+$. If $(s, x) \in C$ we say that s is the *source symbol* and x is the *code word*. If C is an injective function we say that the code is *nonsingular*.

Nonsingularity allows us to unambiguously describe the individual symbols of the source alphabet. For the rest of this book, whenever we talk about a code we mean a nonsingular code. Moreover, without any loss of generality, we will restrict ourselves to *binary codes*, that is, $\mathcal{X} = \mathcal{B} = \{0, 1\}$.

The property of nonsingularity can also be applied to strings of symbols. In order to do that, we have to extend the concept of code from symbols to strings.

Definition 5.1.2 The *extension of order n* of a code C is a function $C^n : \mathcal{S}^n \rightarrow \mathcal{B}^+$ defined as $C^n(s_{i_1} \dots s_{i_n}) = C(s_{i_1}) \dots C(s_{i_n})$, where $C(s_{i_1}) \dots C(s_{i_n})$ is the concatenation of the code words corresponding to the symbols of the string $s_{i_1} \dots s_{i_n} \in \mathcal{S}^n$. An extension of order n of a code C is *nonsingular* if the function C^n is injective.

If it is clear from the context, we will also use the word *code* to refer to a nonsingular extension of order n of a code, and the elements of \mathcal{S}^n will be called *source words*.

■ **Example 5.1** The code $C(a) = 0$, $C(b) = 00$, $C(c) = 01$, and $C(d) = 11$ is a nonsingular code, but its extension of order 2 is singular, since, for example, $C^2(ab) = C^2(ba) = 000$. ■

As we have seen in Example 5.1 not all nonsingular codes have nonsingular extensions, that is, it might happen that we are not able to decode the original messages given their encoded versions. Unique decodability is a highly desirable property of codes.

Definition 5.1.3 A code C is called *uniquely decodable* if its order n extension C^n is nonsingular for all n .

Next proposition provides a characterization of the unique decodability of codes.

Proposition 5.1.1 A code C is uniquely decodable if, and only if, the function $C^+ : \mathcal{S}^+ \rightarrow \mathcal{B}^+$ is injective.

Proof. If the function C^+ is injective, then the restriction to C^n must be injective for all n . Now, let us assume that C^n is nonsingular for all n and let's prove that C^+ must be nonsingular by contradiction: select two source words $s_1 \in \mathcal{S}^n$ and $s_2 \in \mathcal{S}^m$, $n \neq m$, and assume that they have the same code word $C^+(s_1) = C^+(s_2)$. Then construct the words $s_3 = s_1s_2$ and $s_4 = s_2s_1$, both s_3 and s_4 have the same length, and $C^+(s_3) = C^+(s_4)$, which is a contradiction with the fact that C^{n+m} must be nonsingular. ■

■ **Example 5.2** The code $C(a) = 0$, $C(b) = 01$, $C(c) = 011$, and $C(d) = 0111$ is a uniquely decodable code. For example, the code word 0010011 uniquely corresponds to the source word *abac*. Unique decodability is achieved because the 0 symbol plays the role of a delimiter, separating code words. ■

The Sardinas-Patterson theorem provides a necessary and sufficient condition for a code to be uniquely decodable. The theorem is based on an algorithmic approach, enabling the examination of a code's unique decodability through a sequence of iterative steps. Let C_0 denote the set of code words of a code C . We define the set C_n as

$$C_n = C^{-1}C_{n-1} \cup C_{n-1}^{-1}C$$

for all $n \in \mathbb{N}$ and where $C^{-1}C_{n-1}$ represents the left quotient of C and C_{n-1} . And let C_∞ be defined as

$$C_\infty = \bigcup_{n=1}^{\infty} C_n$$

Proposition 5.1.2 — Sardinas-Patterson. A code C is uniquely decodable if and only if the sets C_0 and C_∞ are disjoint.

Proof. TODO: Pending ■

■ **Example 5.3** Given the code of Example 5.1 we have that

$$C_0 = \{0, 00, 01, 11\}$$

$$C_1 = \{0, 1\}$$

And since $C_0 \cap C_1 \neq \emptyset$ we can conclude that the code is not uniquely decodable.

Given the code of Example 5.2 we have that

$$C_0 = \{0, 01, 011, 0111\}$$

$$C_1 = \{1, 11, 111\}$$

$$C_2 = \emptyset$$

And since $C_0 \cap C_\infty = C_0 \cap \bigcup_{n=1}^{\infty} C_n = C_0 \cap C_1 = \emptyset$ we can conclude that the code is uniquely decodable. ■

Next definition introduces the concept of prefix-free codes. Prefix-free codes will play a critical role in the computation of the amount of algorithmic information of an arbitrary string (described in Chapter 6), and in our own theory of nescience. Prefix-free codes also allow us to link coding theory and probability theory through the Kraft inequality (Theorem 5.2.1). Note that we prefer the name *prefix-free code* over the more standard *prefix code*, since the former more accurately describes the concept.

■ **Definition 5.1.4 — Prefix-free Code.** A code C is *prefix-free* if for all i, j where $1 \leq i, j \leq q$ and $i \neq j$, $C(s_i)$ is not a prefix of $C(s_j)$.

Note that the fact that a string is a prefix of itself does not violate the prefix-free property because the condition specifically excludes considering a string as a prefix of itself ($i \neq j$) for the purpose of determining whether a set of code words is prefix-free.

■ **Example 5.4** The code $C(a) = 0$, $C(b) = 10$, $C(c) = 110$ and $C(d) = 1110$ is a prefix-free code. Here, the 0 symbol plays the role of a comma as it was the case of Example 5.2, but its new position at the end of the code words is what makes the code prefix-free. ■

Prefix-free codes are uniquely decodable, as the next proposition proves.

Proposition 5.1.3 Let C be a prefix-free code, then C is uniquely decodable.

Proof. Let C be a prefix-free code, and assume that C is not uniquely decodable. This implies there exist two different sequences of source symbols, say r and s , such that they encode to the same sequence of code words: $C(r) = C(s)$. Since C is prefix-free, the start of any code word sequence uniquely determines the first code word, as no other code word can be a prefix. Therefore, the first symbols in r and s must be the same, as they are encoded into the same first code word. However, this leads to the conclusion that $r = s$, contradicting our initial assumption that r and s are different. Therefore, our assumption must be false, and the code C must be uniquely decodable. ■

From an engineering point of view it is highly convenient to have codes whose source symbols can be decoded as soon as the corresponding code words are received, that is, it is not necessary to wait for the next code word in order to decode the current symbol, this is why some authors refer to these codes as *instantaneous codes*.

Definition 5.1.5 A code C is *instantaneous* if, for any order n and for any sequence of code words $C(s_{i_1}), C(s_{i_2}), \dots, C(s_{i_n})$, each sequence of code words $\mathbf{t} = C(s_{i_1})C(s_{i_2})\dots C(s_{i_m})\dots$ can be uniquely decoded as $\mathbf{s} = s_{i_1}s_{i_2}\dots s_{i_m}\dots$ without ambiguity, regardless of the continuation of \mathbf{t} .

■ **Example 5.5** The code described in Example 5.2 is not instantaneous. For example, after receiving the sequence 011 the source symbol could be a c if the next symbol is a 0 or a d if it is a 1. ■

Prefix-free codes and instantaneous codes are essentially two terms for the same concept. A prefix-free code is one in which no code word is a prefix of any other code word. This property ensures that there is a clear demarcation between code words when they are concatenated in a sequence, allowing each code word to be decoded immediately upon receipt without the need to look ahead to subsequent code words for context.

Proposition 5.1.4 A code C is instantaneous if, and only if, it is a prefix code.

Proof. Assume C is an instantaneous code, and suppose that C is not a prefix-free code. This means there exists at least a pair of code words, say $C(s_i)$ and $C(s_j)$, such that $C(s_i)$ is a prefix of $C(s_j)$. Consider a sequence where $C(s_j)$ is transmitted. Since $C(s_i)$ is a prefix of $C(s_j)$, the decoder would decode $C(s_i)$ from the initial part of $C(s_j)$, leading to ambiguity, as the rest of $C(s_j)$ could be seen as another code or part of the next code. This contradicts the assumption that C is instantaneous.

Assume C is a prefix code. This property ensures that once a code word is identified in a sequence, there can be no confusion as to where it ends, and the next code word begins. There is no need to look ahead to determine the boundary between code words, as the end of one code word cannot be mistaken for the beginning of another. Hence, a prefix code allows for instantaneous decoding. ■

The last type of codes we are going to review are fixed length codes. We will use fixed length codes to compute the length of a text when there are no regularities we can exploit to compress it.

Definition 5.1.6 If all the code words of a code have the same length we say that the code is a *fixed length code*.

Fixed length codes have the property of being prefix-free (or instantaneous).

Proposition 5.1.5 Let C be a fixed-length code, then C is prefix-free.

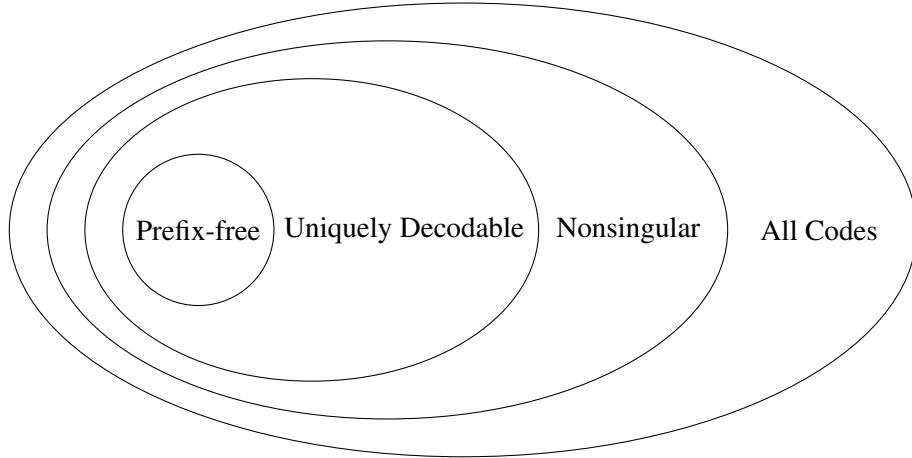


Figure 5.1: Classification of Codes

Proof. Let C be a fixed length code, and $C(s_i)$ and $C(s_j)$ the code words of two arbitrary source words s_i and s_j . Assume that $C(s_i) <_p C(s_j)$, given the fact that $l(C(s_i)) = l(C(s_j))$ we have that $C(s_i) = C(s_j)$ and so, the code C is prefix-free. ■

Of course, the converse of the previous proposition does not hold; not all prefix-free codes are of fixed length. The code described in Example 5.4 is prefix-free but it is not fixed-length.

Figure 5.1 provides a graphical representation of the relation among the different types of codes that have been introduced in this section.

5.2 Kraft Inequality

The Kraft inequality provides a condition for the existence of a prefix-free code given a set of code word lengths. Kraft's inequality states that for a given set of code word lengths in a binary code, the sum of the reciprocals of the powers of two corresponding to the code word lengths must be less than or equal to one. This condition is both necessary and sufficient; not only must any prefix-free code satisfy this inequality, but given a set of lengths that meets this condition, it is always possible to construct a corresponding prefix-free code. The elegance and utility of Kraft's inequality lie in its ability to link the lengths of code words with the mathematical properties of probability.

Theorem 5.2.1 — Kraft Inequality. Let $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$ be a set of lengths, where $l_i \in \mathbb{N}$, then there exists a binary prefix-free code C whose code words have the lengths of \mathcal{L} if, and only if,

$$\sum_{l_i \in \mathcal{L}} 2^{-l_i} \leq 1$$

Proof. Consider a binary tree whose branches are labeled with the symbols of the code alphabet, in such a way that the path from the root to the leaves traces out the symbols of a code word. The prefix-free condition implies that nodes representing complete code words cannot have descendants. An example of such a tree, for the code described in Example 5.4, is shown in Figure 5.2.

Let $l_{max} = \max \{l_1, l_2, \dots, l_q\}$, that is, the length of the longest code word from the set of lengths. There will be at most $2^{l_{max}}$ leaf nodes in the tree, but at level l_i we have to prune $2^{l_{max}-l_i}$ leaves, since the code is prefix-free. Summing over all the code words' lengths, we have that the total number of

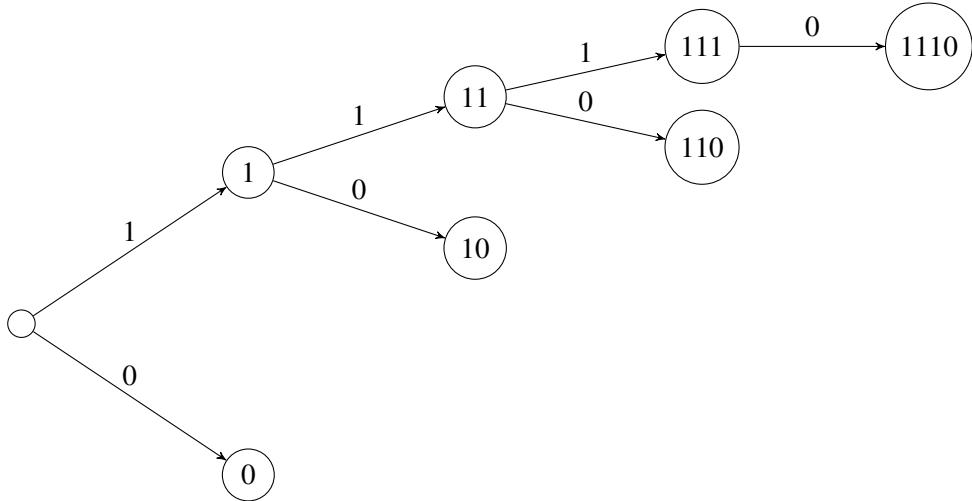


Figure 5.2: Prefix-free Tree

pruned leaves must be less than or equal to the maximum number of leaves, that is

$$\sum_{l_i \in \mathcal{L}} 2^{l_{max} - l_i} \leq 2^{l_{max}}$$

or, equivalently,

$$\sum_{l_i \in \mathcal{L}} 2^{-l_i} \leq 1$$

which is exactly the inequality we are trying to prove.

Conversely, given any set of code words' lengths $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$ that satisfy the Kraft inequality, we can always construct a binary tree, like the one in the Figure 5.2. Label the first node (lexicographically) of depth l_1 as code word 1, and remove its descendants from the tree. Then label the first remaining node of depth l_2 as code word 2, and so on. Proceeding this way, we construct a prefix code with the specified lengths. ■

Given a code C whose code word lengths \mathcal{L} satisfy the Kraft inequality does not necessarily mean that the code is prefix-free, since what the inequality states is that there exists a prefix-free code with those word lengths, not that all codes with those word lengths are prefix-free.

■ **Example 5.6** The code $C(a) = 0$, $C(b) = 111$, $C(c) = 110$ and $C(d) = 100$ satisfies the Kraft inequality, but it is not prefix-free. ■

McMillan's Inequality enriches our understanding of coding theory by establishing a crucial link between the lengths of code words in uniquely decodable codes. This inequality mirrors Kraft's Inequality, yet it broadens the scope to encompass all uniquely decodable codes, not just those that are prefix-free. By demonstrating that the sum of the reciprocals of the powers of two for code word lengths must be less than or equal to one for a code to be uniquely decodable, McMillan's Inequality ensures that such codes can be constructed with a given set of lengths.

Theorem 5.2.2 — McMillan's Inequality. Let $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$ be a set of lengths, where $l_i \in \mathbb{N}$, then there exists a uniquely decodable code C whose code words have the lengths of \mathcal{L} if,

and only if,

$$\sum_{l_i \in \mathcal{L}} 2^{-l_i} \leq 1$$

Proof. TODO: review

We must show that if there exists a uniquely decodable code with code word lengths l_1, l_2, \dots, l_q , then the inequality holds.

Suppose C is a uniquely decodable code with code word lengths l_1, l_2, \dots, l_q . Consider a sequence of code words produced by C that is long enough to contain any possible concatenation of n code words (for some n large enough). Let's denote the set of all such sequences as S_n .

Because C is uniquely decodable, each sequence in S_n corresponds to a unique sequence of source symbols. The number of different sequences in S_n is at least 2^{nl} , where l is the smallest length among all l_i . This is because, for the shortest code word, we can consider a sequence of length nl and all possible fillings with the shortest code word. Since the code is uniquely decodable, none of these sequences can be the same as any sequence containing longer code words.

The total number of bits required to represent all sequences in S_n can indeed be calculated as $(2^{-l_1} + 2^{-l_2} + \dots + 2^{-l_q})^n$, considering the combinatorial possibilities of all code word concatenations.

Since each sequence in S_n must be different (uniquely decodable) and can be represented in a binary tree (where each leaf corresponds to a code word and the length of the path to the leaf corresponds to the length of the code word), the number of sequences in S_n cannot exceed $2^{nl_{max}}$, where l_{max} is the length of the longest code word.

Therefore, we have:

$$(2^{-l_1} + 2^{-l_2} + \dots + 2^{-l_q})^n \leq 2^{nl_{max}}$$

Taking the n th root of both sides, we have:

$$2^{-l_1} + 2^{-l_2} + \dots + 2^{-l_q} \leq 2^{l_{max}}$$

Since $2^{l_{max}}$ is always less than or equal to 1 (because there must be at least one code word of the maximum length and thus occupying the "full" length of the binary tree), we obtain:

$$2^{-l_1} + 2^{-l_2} + \dots + 2^{-l_q} \leq 1$$

To prove sufficiency, we must show that if the inequality holds, then there exists a uniquely decodable code with those lengths.

If the inequality holds, we can use the Kraft construction to build a prefix-free code, which is always uniquely decodable, with code word lengths l_1, l_2, \dots, l_q . The construction starts by assigning the shortest code word first and ensuring no other code word is a prefix of any other. Since prefix-free codes are a subset of uniquely decodable codes, we have constructed the required code.

Therefore, by satisfying the inequality, we can construct a uniquely decodable code, proving that the condition is sufficient. ■

In the theory of nescience, we are primarily interested in prefix-free codes. This preference may seem to impose a limitation, as it appears more rational to consider the broader category of uniquely

decodable codes. However, this perceived limitation does not actually exist. As the following theorem demonstrates, uniquely decodable codes also satisfy Kraft's inequality. This means that for any uniquely decodable code, there exists a prefix-free code with exactly the same code word lengths, ensuring no compromise in the generality of our analysis when focusing on prefix-free codes.

Corollary 5.2.3 There is an instantaneous (prefix-free) code with word lengths l_1, \dots, l_q if and only if there is a uniquely decodable code with these word lengths.

Proof. **TODO: Review** This corollary is a direct consequence of McMillan's Inequality, which is a counterpart to Kraft's Inequality for uniquely decodable codes. McMillan's Inequality states that for a set of code word lengths $\{l_1, l_2, \dots, l_q\}$, a uniquely decodable code exists if and only if the sum of 2^{-l_i} for all i is less than or equal to one. Given that prefix-free codes are a subset of uniquely decodable codes and also satisfy this condition, the existence of a uniquely decodable code with given word lengths implies the possibility of constructing a prefix-free code with the same lengths. The reverse is inherently true by the definition of prefix-free codes, which are always uniquely decodable. ■

In the context of nescience, our interest lies not in the specific codes themselves but in the lengths of the code words. This emphasis allows us to abstract away from the details of code construction to concentrate on the mathematical properties and implications of these lengths, which are central to understanding and applying the principles of nescience.

5.3 Optimal Codes

In coding theory, a compact code is an optimal encoding strategy that minimizes the expected length of code words, a concept central to evaluating a code's efficiency. The expected length of a code is determined by the weighted average of the lengths of its code words, with weights corresponding to the probability distribution of the source symbols. By designing code words so that more frequent symbols are assigned shorter lengths and less frequent ones longer lengths, compact codes effectively reduce the average size needed to encode information. This principle is pivotal in data compression, as it allows for a significant reduction in the space required for storage or the bandwidth needed for transmission.

Let's establish $\mathcal{S} = \{s_1, s_2, \dots, s_q\}$ as a finite source alphabet, with P denoting a probability distribution defined over the elements of \mathcal{S} .

Definition 5.3.1 The *expected length* of a code C , denoted by L_C , is the weighted sum of the lengths of the code words, calculated as

$$L_C = \sum_{i=1}^q P(s_i)l_i,$$

where $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$ represents the lengths of the code words of C . We may simply use L to denote L_C when the code C is understood from the context.

Our goal is to identify a code C that minimizes the expected length L of the codewords \mathcal{L} under the given probability distribution P . Such an optimal code C would enable us to compress messages composed of the source alphabet \mathcal{S} effectively, reducing the number of symbols required to encode these messages.

Next, the following definition formalizes what it means for a code to be compact, which is a desirable attribute for efficient data encoding.

Definition 5.3.2 A code C is *compact* if its average length L is less than or equal to the average length of all the other codes for the same source alphabet and code alphabet.

Definition 5.3.3 A code C is *compact* if, for a given source alphabet \mathcal{S} and a corresponding probability distribution, its expected length L_C is minimized among all possible codes that can be constructed for \mathcal{S} .

This means C achieves the lowest possible weighted average codeword length, where each codeword length is weighted by the probability of its corresponding source symbol.

The existence of compact codes for all possible source alphabets is a foundational aspect of coding theory, suggesting that for every finite source alphabet, an optimally efficient encoding scheme can be devised. It is not entirely clear that compact codes exist for all possible source alphabets, so the following proposition must be proven.

Proposition 5.3.1 For every finite source alphabet $\mathcal{S} = \{s_1, s_2, \dots, s_q\}$ with a defined probability distribution over its symbols, there exists at least one code C that is compact.

Proof. The proof of this proposition involves constructing or identifying a code C for the source alphabet \mathcal{S} that achieves the minimum possible expected length L_C . This can be demonstrated through the use of Huffman's algorithm, which is designed to produce an optimal prefix code based on the probabilities of the source symbols. By definition, the Huffman code for a given probability distribution over \mathcal{S} minimizes the expected length of the codewords, thereby proving the existence of a compact code for any finite source alphabet with a given probability distribution. ■

This clarification not only completes the explanation but also aligns the definition of compactness with established principles in coding theory. It provides a more rigorous basis for discussing the efficiency of encoding schemes and underscores the role of probability distributions in determining the compactness of a code.

TODO: Introduce this concept

Definition 5.3.4 The redundancy of a code is defined as

$$\eta = \frac{H((S))}{L}$$

Of course, our goal is to minimize the redundancy of codes.

Next theorem states that the entropy of the probability distribution P poses a limit to the average length of prefix-free codes.

Theorem 5.3.2 The expected length L_C of any prefix-free r -ary code, given the probability distribution P , is greater than or equal to the entropy of P , that is

$$H_r(P) \leq L_C$$

with equality if, and only if, $r^{-l_i} = P_i$ for all $0 \leq i \leq q$.

Proof.

Definition 5.3.5 A probability distribution is called *D-adic* if each of the probabilities is equal to D^n for some n .

Corollary 5.3.3 We have the equality in the theorem if, and only if, the distribution of X is D-adic.

Proof. TODO

Mention that in practice we will non-integer code word lengths. Show that, on average, the length of the encoded string will be less than 1 bit than using a code with code words with integer lengths

Kraft's inequality allows us to compare how efficient different codes are for the same source alphabet.

Definition 5.3.6 Let $C_1 : \mathcal{S} \rightarrow \mathcal{X}^+$ and $C_2 : \mathcal{S} \rightarrow \mathcal{X}^+$ be two different codes. We say that code C_1 is *more efficient* than code C_2 if for all $s \in \mathcal{S}$ we have that $l(C_1(s)) \leq l(C_2(s))$, and there exists at least one $s' \in \mathcal{S}$ such that $l(C_1(s')) < l(C_2(s'))$.

■ **Example 5.7** The code described in Example 5.6 is more efficient than the code of Example 5.4. Of course, the problem with the code described in Example 5.6 is that it is not prefix-free, but since it satisfies Kraft's inequality, we know that there must exist another code with the same code word lengths that is prefix-free. For example, $C(a) = 0$, $C(b) = 10$, $C(c) = 110$ and $C(d) = 111$. ■

We are interested in the most efficient possible codes.

TODO: This definition is wrong

Definition 5.3.7 A code C is *complete* if there does not exist a code C' that is more efficient than C .

It turns out that Kraft's inequality provides a very useful characterization of complete codes.

Proposition 5.3.4 A code C is complete if, and only if, its code word lengths $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$ satisfy the property:

$$\sum_{l_i \in \mathcal{L}} 2^{-l_i} = 1$$

Proof. Suppose the sum of the exponentiated inverses of the code word lengths equals 1:

$$\sum_{l_i \in \mathcal{L}} 2^{-l_i} = 1$$

This implies that the code utilizes the full capacity of the coding space. In other words, there is no redundancy or additional space in the code that could be used to encode a symbol with a shorter length. This is because each term in the sum 2^{-l_i} represents the proportion of the coding space taken up by the code word of length l_i . If these proportions sum up to 1, then all possible code words of the given lengths are used, and there's no space left to introduce any new code word without increasing the length of at least one existing code word. Therefore, the code is as efficient as possible, hence complete.

Now suppose that the code C is complete. This means that there is no other code C' with the same source alphabet and the same or shorter code word lengths that is more efficient than C .

Assume for contradiction that the sum of the exponentiated inverses is less than 1:

$$\sum_{l_i \in \mathcal{L}} 2^{-l_i} < 1$$

This would imply that there is still unused coding space, meaning that it would be possible to construct a more efficient code by using this unused space to represent at least one symbol with a shorter code word, contradicting the assumption that C is complete. Therefore, the sum cannot be less than 1.

Since C is complete and cannot be less efficient, the only alternative left is that the sum is exactly 1:

$$\sum_{l_i \in \mathcal{L}} 2^{-l_i} = 1$$

Combining both directions, we have shown that a code C is complete if and only if the sum of the exponentiated inverses of its code word lengths equals 1, as stated in the proposition. ■

5.4 Entropy

In this section we are going to introduce the concept of *entropy*, as a measure of the uncertainty of a random variable. Entropy is a very difficult to grasp concept that can be applied in many different contexts, such as communications, statistics, finance, etc. Here we are interested in entropy because it will allow us to identify codes with the shortest possible average length.

Let $A = \{a_1, a_2, \dots, a_n\}$ a finite set, and X a random variable defined over the set A with probability mass function $p(a)$.

Definition 5.4.1 — Entropy. The *entropy* of the random variable X , denoted by $H(X)$ and measured in *bits*, is defined as:

$$H(X) = \sum_{a \in A} p(a) \log \frac{1}{p(a)}$$

Note that the entropy of X does not depend on the individual elements of A , but on their probabilities. It is easy to show that $H(X) \geq 0$ since $0 \leq p(a) \leq 1$ implies that $-\log p(a) \geq 0$. In case of $p(a_i) = 0$ for some i , the value of the corresponding summand $0 \log 0$ is taken to be 0, which is consistent with the limit $\lim_{p \rightarrow 0^+} p \log p = 0$. If we change the base of the logarithm to u , entropy will be scaled by a factor of $\log_u 2$ (see Equation C.1).

■ **Example 5.8** Let X a random variable defined over the set $A = \{a_1, a_2\}$, with values $p(a_1) = q$ and $p(a_2) = 1 - q$. Then, the entropy of X is given by:

$$H(X) = q \log \frac{1}{q} + (1 - q) \log \frac{1}{1 - q}$$

Figure 5.3 shows the entropy of X for different values of q . If $q = 0$ or $q = 1$ the entropy is 0, that is, there is no uncertainty about which value of A we will get. The maximum value of H is 1, and it is reached when $q = 1/2$; that is, we could say that 1 bit is the uncertainty associated to two equally probable symbols. ■

Next proposition shows that the maximum value for entropy is the logarithm of the number of symbols of A , and that this value is reached when all the symbols have the same probability.

Proposition 5.4.1 Given the random variable X we have that $H(X) \leq \log n$, and $H(X) = \log n$ if, and only if, $p(a_1) = p(a_2) = \dots = p(a_n)$.

Proof. Consider the expression:

$$\log n - H(X) = \sum_{i=1}^n p(a_i) \log n - \sum_{i=1}^n p(a_i) \log \frac{1}{p(a_i)} = \sum_{i=1}^n p(a_i) \log np(a_i)$$

Applying property C.1 we have that:

$$\log n - H(X) = \log e \sum_{i=1}^n p(a_i) \ln np(a_i)$$

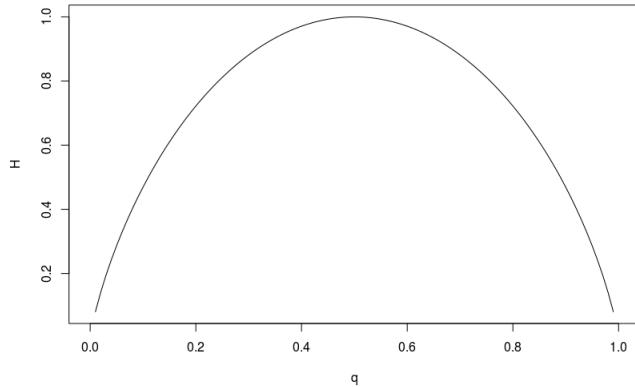


Figure 5.3: Binary Entropy Function

And applying property C.2 (equalling $x = 1/n p(a_i)$):

$$\log n - H(X) \geq \log e \sum_{i=1}^n p(a_i) \left(1 - \frac{1}{np(a_i)} \right) \geq \log e \left(\sum_{i=1}^n p(a_i) - \frac{1}{n} \sum_{i=1}^n \frac{p(a_i)}{p(a_i)} \right) \geq 0$$

Which proves that $H(X) \leq \log n$.

The inequality becomes an equality if, and only if, $p(a_i) = 1/n$ (given that the inequality C.2 becomes an equality if, and only if, $x = 1$). ■

■ Example 5.9 If we choose a random symbol from A according to the probability mass function p , entropy would be the minimum expected number of binary questions (Yes/No questions) required to identify the selected symbol. If the symbols of A are equiprobable, the expected number of questions is maximal and equal to $\log d(A)$. ■

We can extend the concept of entropy to a pair of random variables by means of using the joint probability mass function. In this way, the joint entropy will be a measure of the uncertainty associated to both variables. Let $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_m\}$ two finite sets, and X and Y two random variables defined over the sets A and B respectively, with probability mass function $p(a)$ and $p(b)$, and joint probability mass function $p(a, b)$.

Definition 5.4.2 The *joint entropy* of the random variables A and B , denoted by $H(A, B)$, is defined as:

$$H(A, B) = \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(a, b)}$$

Since $p(a, b) = p(b, a)$ we have that the joint entropy does not depend of the order in which the random variables are selected, that is $H(A, B) = H(B, A)$. We can provide a similar definition for the joint entropy of a set of n random variables A_1, A_2, \dots, A_n using the joint probability mass function $p(a_1, a_2, \dots, a_n)$.

Adding a second random variable whose outcome is not known might increase the entropy, as following proposition proves.

Proposition 5.4.2 We have that

$$H(A, B) \geq \max(H(A), H(B))$$

Proof.

$$H(A, B) = \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(a, b)} \geq \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(a)} = \sum_{a \in A} p(a) \log \frac{1}{p(a)} = H(A)$$

In the same way we can prove that $H(A, B) \geq H(B)$. Combining both inequalities we get the desired result. ■

The joint entropy of two random variables cannot be greater than the sum of their individual entropies.

Proposition 5.4.3 We have that $H(A, B) \leq H(A) + H(B)$ and $H(A, B) = H(A) + H(B)$ if, and only if, $p(a)$ and $p(b)$ are statistically independent.

Proof. TODO: Prove without using the concept of conditional entropy nor mutual information. ■

The next derived concept from entropy that we are going to introduce is conditional entropy. Conditional entropy measures the uncertainty of a random variable given that the value of another random variable is known.

Definition 5.4.3 The *conditional entropy* of the random variable B given the random variable A , denoted by $H(B | A)$, is defined as:

$$H(B | A) = \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(b | a)}$$

Since $p(b | a) \neq p(a | b)$ we have that $H(B | A) \neq H(A | B)$. If $H(B | A) = 0$ we have that the value of B is completely determined by the value of A .

Next proposition proves that knowing the value of a second random variable can never increase the uncertainty of a random variable.

Proposition 5.4.4 Given the random variables X and Y , we have that $H(Y | X) \leq H(Y)$, and $H(Y | X) = H(Y)$ if, and only if, $p(a)$ and $p(b)$ are independent.

Proof.

$$\begin{aligned} H(Y | X) &= \sum_{a \in A} \sum_{y \in B} p(a, y) \log \frac{1}{p(y | a)} = \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{p(a)}{p(a, b)} \\ &= \sum_{a \in A} \sum_{b \in B} p(a, b) \log p(a) - \sum_{a \in A} \sum_{b \in B} p(a, b) \log p(a, b) = -H(X) + H(X, Y) \end{aligned}$$

Applying Proposition 5.4.3 we have that $H(Y | X) = H(X, Y) - H(X) \leq H(X) + H(Y) - H(X) = H(Y)$. The iff equality is also proved by applying Proposition 5.4.3. ■

From an intuitive point of view we could expect that the uncertainty associated to a pair of random variables must be equal to the uncertainty of one of them plus the uncertainty of the second given that we know the outcome of the first one.

Proposition 5.4.5 — Chain rule. Given the random variables X and Y we have that $H(X, Y) = H(X) + H(Y | X)$.

Proof.

$$\begin{aligned} H(Y, X) &= \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(a, b)} = \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(a)p(a | b)} \\ &= \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(a)} + \sum_{a \in A} \sum_{b \in B} p(a, b) \log \frac{1}{p(a | b)} = H(X) + H(Y | X) \end{aligned}$$

■

The last derived concept of entropy we are going to see is mutual information. Intuitively, the mutual information of two random variables X and Y measures the information that X and Y share, that is, how much knowing one of these variables reduces the uncertainty about the other.

Definition 5.4.4 The *mutual information* of the random variable X and Y , denoted by $I(X;Y)$, is defined as:

$$I(X;Y) = \sum_{a \in A} \sum_{b \in B} p(a,b) \log \frac{p(a,b)}{p(a)p(b)}$$

Since $p(a,b) = p(b,a)$ we have that $I(X;Y) = I(Y;X)$, that is, the order of the random variables does not affect the concept of mutual information.

Next proposition shows that mutual information is a positive quantity, and it is equal to 0 if, and only if, the random variables are independent.

Proposition 5.4.6 Given the random variables X and Y we have that $I(X;Y) \geq 0$, and $I(X;Y) = 0$ if, and only if, the variables X and Y are independent.

Proof. TODO: to be done ■

Introduce this proposition

Proposition 5.4.7 Given the random variables X and Y , we have that:

$$I(X;Y) = H(X) - H(X | Y) = H(Y) - H(Y | X)$$

Proof. TODO: to be done ■

Introduce this proposition

Proposition 5.4.8 Given the random variables X and Y , we have that:

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

Proof. TODO: to be done ■

Prove that $I(\mathcal{S};\mathcal{S}) = H(\mathcal{S})$

Use the Venn diagrams in this example

■ Example 5.10

5.5 Huffman Algorithm

This section should be about compression algorithms. Not sure if only about algorithms based on information theory, or generic compression algorithms. Depends of what we need in practice

Mention that Huffman is not necessarily the optimal compression algorithm

From a practical point of view, there exists an algorithm, called *Huffman algorithm*, that provides a method to build compact prefix-free codes given a probability distribution. For simplicity, we will study first the particular case of constructing binary prefix-free codes, and later I will provide its generalization to the case of D-ary prefix-free codes.

The algorithm (see Algorithm 5.1) expects as input a source alphabet $\mathcal{S} = \{s_1, s_2, \dots, s_q\}$ and their corresponding probabilities $P = \{p_1, p_2, \dots, p_q\}$. For simplicity, we will merge both sets into a single one $Q = \{(s_1, p_1), (s_2, p_2), \dots, (s_q, p_q)\}$. The algorithm works by constructing a binary tree T , similar to the one used in the proof of Theorem 5.2.1. The algorithm requires $d(Q) - 1$

Algorithm 5.1 Huffman Algorithm

```

procedure HUFFMAN( $Q$ )
     $T \leftarrow$  empty tree
    for  $i \leftarrow 1, d(Q) - 1$  do
        allocate a new node  $z$ 
         $z.\text{left} = x = \text{EXTRACT-MIN}(Q)$ 
         $z.\text{right} = y = \text{EXTRACT-MIN}(Q)$ 
         $z.\text{freq} = x.\text{freq} + y.\text{freq}$ 
         $\text{INSERT}(Q, z)$ 
    end for
    return  $T$ 
end procedure

```

iterations to finish. During each iteration, the two elements with the lowest probability are selected and removed from set Q , and a new tree node z is created, with the addition of the removed values, and added to the set Q . Once the tree has been constructed, we have to perform a tree transversal assigning a 0 to each left branch, and a 1 to each right branch, until we reach a leaf.

■ **Example 5.11** Assume we have the source alphabet $S = \{a, b, c, d, e, f\}$ with the associated probabilities $P = \{0.35, 0.16, 0.08, 0.12, 0.06, 0.23\}$. In Figure are depicted the contents of the set Q and the tree T for each iteration of the algorithm. At the end of the algorithm, if we perform a traversal of the T tree, we will get the following prefix-free compact code for the source alphabet S :

Source Word	Code Word
a	11
b	00
c	1011
d	100
e	1010
f	01

The expected length of the code is $L = 2.4$, and its entropy is $H \approx 2.34$. Since the set of probabilities is not D-adic ... ■

This order is arbitrary; switching the left and right child of any node yields a different code of the same cost

Proposition 5.5.1 Given the probability is ... ■

Proof. TODO ■

The next theorem shows the optimality of the Huffman coding.

Theorem 5.5.2 If C is a Huffman code then C is compact. ■

Proof. TODO ■

TODO: Rewrite the following paragraphs

So not only is this code optimal in the sense that no other feasible code performs better, but it is very close to the theoretical limit established by the entropy

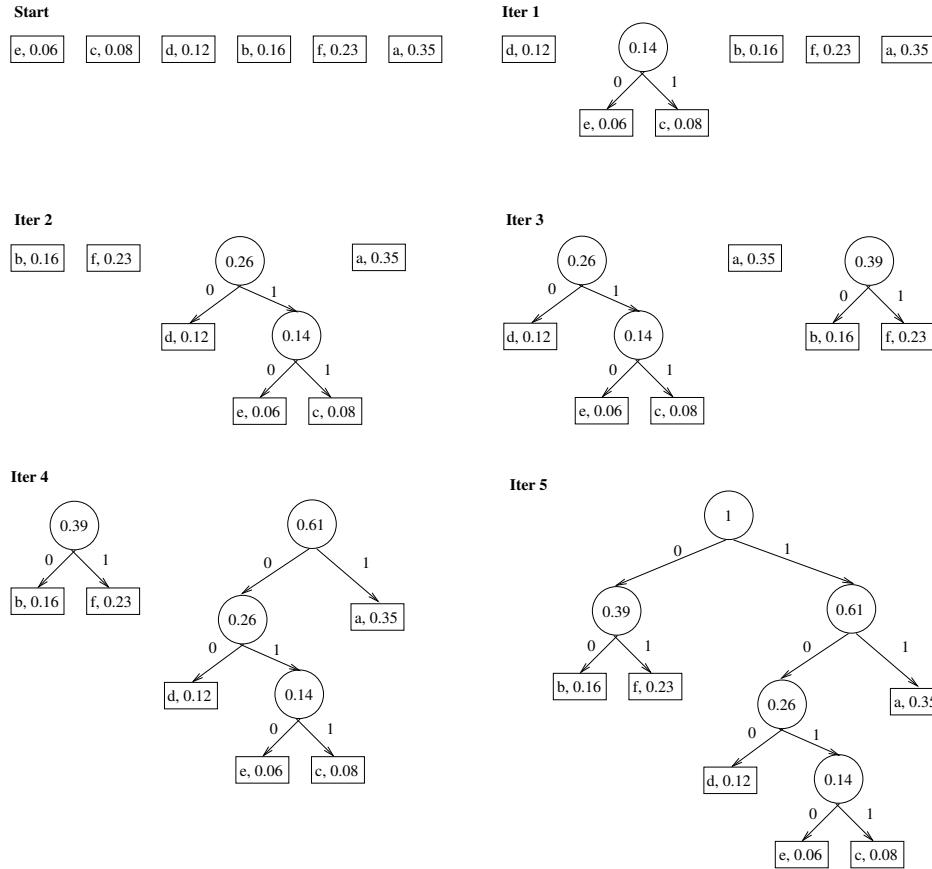


Figure 5.4: Huffman Algorithm

Although we have proved the theorem for a binary alphabet, the proof can be extended to establishing optimality of the Huffman coding algorithm for a D-ary alphabet as well.

TODO: show how to extend the algorithm to D-ary codes

Then -ary Huffman algorithm uses the $0, 1, \dots, n-1$ alphabet to encode message and build an n -ary tree [...] the same algorithm applies as for binary codes, except that the n least probable symbols are taken together, instead of just the 2 least probable. Note that for n greater than 2, not all sets of source words can properly form an n -ary tree for Huffman coding. In this case, additional 0-probability place holders must be added. This is because the tree must form and n to 1 contractor; for binary coding, this is a 2 to 1 contractor, and any sized set can form such a contractor. If the number of source words is congruent to 1 modulo $n-1$, then the set of source words will form a proper Huffman tree.

Mention arithmetic coding

5.6 Discretization Algorithms

TODO: Rewrite this section.

Let \mathcal{X} a continuous random variable that follows a probability density function $P_{\mathcal{X}}$, and assume we have collected n independent and identically distributed samples $\mathbf{x} = \{x_1, \dots, x_n\}$ from \mathcal{X} . We are interested in computing the length of a compressed version of \mathbf{x} using an optimal compressor. Unfortunately, and except for some degenerate distributions, there is no lossless compression algorithm that produces a string with fewer bits than encoding directly the elements \mathbf{x} . Compression algorithms for continuous data only work in case that the elements of \mathbf{x} are not independent, as it is

the case with images or sound. But, if this is not the case, the only option available to compress \mathbf{x} is to use a lossy compression algorithm, where some information is lost.

We are looking for an algorithm to produce a finite non-overlapping partition of m discrete intervals $D = \{[d_0, d_1], (d_1, d_2], \dots, (d_{m-1}, d_m]\}$, where $d_0 = \min \mathbf{x}_j$, and $d_m = \max \mathbf{x}_j$, and $d_i < d_{i+1}$ for $i = 0, 1, \dots, m - 1$, assign a unique label to each interval, and encode the elements of \mathbf{x} using this labeling schema. As compression algorithm we will use an optimal length code given the relative frequencies of the labels in the encoded vector. In this sense, our goal is to have a collection of intervals with sufficiently number of samples (so they are statistically significant) and that the distribution of frequencies resembles the original probability distribution $P_{\mathcal{X}}$.

A discretization algorithm is a mapping between a (possibly huge) number of numeric values and a reduced set of discrete values, and so, it is a process in which some information is potentially lost. The choice of discretization algorithm is something that could have a high impact in the practical computation of the nescience. We are interested in a discretization algorithm that produces a large number of intervals (low bias), with a large number of number of observations per interval (low variance). Common techniques include *equal width discretization*, *equal frequency discretization* and *fixed frequency discretization*. However, these techniques require the optimization of an hyperparameter, and so, they are not suitable for our purposes.

In a *proportional discretization approach* the number of intervals m and the number of observations per interval s are equally proportional to the number of observations n . The algorithm starts by sorting the values of \mathbf{x}_j in ascending order and then discretizing them into m intervals of approximately s (possibly identical) values each. In this way, as the number of training observations increases, both interval frequency and number of intervals increases, taking advantage of the larger number of observations. In the same way, when the number of observations decreases, we reduce both.

5.6.1 k-means Clustering

K-means clustering is an algorithms that partitions n observations into k clusters in such a way that each observation belongs to the cluster with the nearest mean. In this way, we can replace the observation that belong to a cluster by its means, as a discretization. The optimization criteria in k-means is to minimize the within-cluster variance. Given the fact that the problem is NP-Complete, some approximation algorithms are used instead.

TODO: Define the concept of Voronoi diagram / Voronoi cell

References

TODO: write this section

In 1948, Claude E. Shannon published a paper entitled "A Mathematical Theory of Communication", where he established the foundations of a new discipline, later called *information theory*.

Paper of Shannon ... Harley

Huffman -> D. A. Huffman. A method for the construction of minimum redundancy codes. Proc. IRE, 40:1098-1101, 1952.

The algorithm of huffman has been adapted from Cover. Here also you can find a proof that the algorithm is of order XX.

Kraft's inequality was published by Leon G. Kraft in 1949 as part of his Master Thesis [[kraft1949device](#)]. The inequality was independently rediscovered and proved for the general case of uniquely decodable codes by Brockway McMillan in 1956 [[mcmillan1956two](#)]. The proofs contained in this book have been adapted from [[cover2012elements](#)].

The proof of proposition 5.4.1 has been adapted from [abramson1963information]. The proof of proposition 4.3.1 has been adapted from Abramson.

In the area of *digital signal processing* [**gersho2012vector**] it is common to apply a pre-processing step called *quantization*, in which a large number of samples from a continuous signal are mapped into a finite number of representative values. It can be a *scalar quantization* when the signal is one dimensional, or *vector quantization* in case of a multidimensional signal. The optimization goal is to identify a (pre-defined) number of quantized values such that the mean squared error between the selected values and the original signal is minimized. The problem is solved using the Lloyd-max algorithm [**lloyd1982least**] (closely related to the kmeans clustering algorithm [**<empty citation>**] used in machine learning) in which the search space is partitioned in a collection of convex regions and their centroids used as quant, and then, are continuously adapted until some stopping criteria is reached. Although it can be shown that the algorithm converges to the optimal solution that minimizes the mean squared error, the selected intervals cannot be used as estimation of the original probability distribution.



6. Complexity

*Everything should be made as simple as possible,
but not simpler.*
Albert Einstein

In Chapter 5, the concept of the complexity of a string based on the lengths of the codewords of a prefix-free code was introduced. This definition is limited by two main factors: firstly, it necessitates prior knowledge of the set of possible strings, and secondly, it requires the definition of a probability distribution over this set *a priori*. It would be highly beneficial if we could expand the set of strings to encompass all strings (that is, \mathcal{B}^*) without necessitating a probability distribution, thereby providing an absolute notion of string complexity. Unfortunately, even if these issues are addressed, a more significant limitation exists in studying the complexity of strings using codes: certain strings, expected to be classified as simple, cannot be compressed. For instance, the binary expansion of the constant π follows a uniform distribution over the set $\{0, 1\}$ and, as such, cannot be compressed. However, it can be fully and effectively described by a very short mathematical formula. This necessitates an alternative definition of string complexity.

Kolmogorov Complexity, also known as *Algorithmic Information Theory*, offers a definition of the complexity of a string that explicitly addresses these issues. Intuitively, the amount of information in a finite string is measured by the length of the shortest computer program capable of producing the string. This approach does not require prior knowledge of the set of valid strings or their probability distribution. Furthermore, objects like π are appropriately classified as having low complexity. We could argue that Kolmogorov complexity provides a universal definition of the amount of information that closely aligns with our intuitive understanding. To compute the Kolmogorov complexity of a string, it is necessary to agree upon a universal description method or computer language, along with a universal computer. One might question whether, by doing so, the complexity of a string becomes dependent on the chosen computer language. Fortunately, it has been demonstrated that this is not the case, as all reasonable (and sufficiently powerful) computer languages yield the same description length, up to a fixed constant that depends on the languages chosen, but not on the string itself. Unfortunately, Kolmogorov complexity introduces a significant

issue: it is a non-computable quantity, and as such, must be approximated in practice.

At this point, one might wonder if it is possible to compute the complexity of any object in general, not just strings. The answer is yes, at least theoretically. Given an object x , the task is to provide an encoding method that represents the object as a string x . This encoding method would only be useful if we can losslessly and effectively reconstruct the original object from its encoded description. Unfortunately, providing such descriptions is not always feasible, either because the objects in question are abstract (as is often the case in mathematics) or because practical reconstruction of the object from its description is currently impossible (for example, with living organisms¹).

6.1 Strings Complexity

In Section 4.1, the concept of the Turing machine, an idealized model of computers, was introduced. We saw that Turing machines can be represented as partial computable functions $T : \mathcal{B}^* \rightarrow \mathcal{B}^*$, which assign to each input string $s \in \mathcal{B}^*$ an output string $T(s) \in \mathcal{B}^*$ (Definition 4.4.1). We also introduced the concept of a universal Turing machine $U : \mathcal{B}^* \times \mathcal{B}^* \rightarrow \mathcal{B}^*$ (Definition 4.2.1), a machine that can simulate the behavior of any other Turing machine; that is, for all $(x, v) \in \mathcal{B}^* \times \mathcal{B}^*$, we have that $U(x, v) = T_x(v)$. Later, in Section 5.1, the concept of a code, and in particular, the notion of a prefix-free code, was introduced (Definition 5.1.4). We saw that this kind of code presents important properties (Theorem 5.2.1). The next definition merges the best of both worlds, Turing machines and prefix-free codes, and introduces a new type of universal Turing machine.

Definition 6.1.1 A *prefix-free universal Turing machine* is a universal Turing machine $U : \mathcal{B}^* \times \mathcal{B}^* \rightarrow \mathcal{B}^*$ such that, for every $v \in \mathcal{B}^*$, the domain U_v is prefix-free, where $U_v : \mathcal{B}^* \rightarrow \mathcal{B}^*$ and $U_v(p) = U(p, v)$ for all $p \in \mathcal{B}^*$.

Using modern computer science terminology we could say that U is the computer, p is the program and v is the input to the program. Intuitively, the above definition requires that no computer program can be a prefix of any other program. This is not a limitation from the point of view of string lengths, since, by applying McMillan's theorem (Theorem 5.2.1), given a uniquely decodable program, we could always find a prefix-free one that computes exactly the same function and has the same length. Moreover, in practice, real computer programs are usually prefix-free; for example, the C programming language requires that all functions must be enclosed by braces {}.

The concept of a prefix-free universal Turing machine allows us to introduce a new definition of the complexity of a string that aligns more closely with our intuitive understanding of the amount of computational information contained in an object (encoded as a string).

Definition 6.1.2 — Kolmogorov Complexity. Fix a prefix-free universal Turing machine $U : \mathcal{B}^* \times \mathcal{B}^* \rightarrow \mathcal{B}^*$. The *Kolmogorov complexity* of a string $s \in \mathcal{B}^*$, denoted by $K(s)$, is defined as:

$$K(s) = \min_{p, v \in \mathcal{B}^*} \{l(p) + l(v) : U(p, v) = s\}$$

Intuitively, the shortest description of a string s is given by two elements: a program p (a self-delimited Turing machine) that compresses all the regular patterns of the string, and a new string v that comprises those parts of s that do not present any regularity. We have to find the optimum balance between increasing the complexity of the program, trying to grasp more regularities, or increase the size of the non-compressible part².

¹As of now, it is not possible to recreate an animal solely based on its DNA.

²In the literature the Kolmogorov complexity of the string s is defined as $K(s) = \min_{p \in \mathcal{B}^*} \{l(p) : U(p, \lambda) = s\}$, that is, the length of the shortest computer program that without any additional input can print the string s . We prefer to use

■ **Example 6.1** Consider the string composed of one thousand times the substring "10", that is " $\underbrace{1010\dots 1010}_{1.000 \text{ times}}$ ". We could write the following program:

```
example(char *v) {
    for (int i=1; i<=1000; i++)
        printf("%s", v);
}
```

and then run it with:

```
example("10");
```

in order to print it. The length of the original string is 2.000 bits, but the length of the program is 480 bits (assuming that every symbol is encoded using an uniform code of 8 bits), and the length of the input is 2 bits, so we can conclude that the string has a low complexity. Of course, in order to compute the real Kolmogorov complexity of the string we should find the shortest Turing machine that prints that string.

On the contrary, a string composed of two thousands random 0's and 1's would have a high complexity, since it does not exists any program shorter than the program that prints the string itself.

■ As we mentioned in the preface of this chapter, Kolmogorov complexity would not be particularly useful if the complexity of strings depended on the choice of universal Turing machine. The following theorem demonstrates that this concern is unfounded, up to a constant that depends on the choice of machines, but not on the strings themselves. This establishes Kolmogorov complexity as an inherent property of the strings.

Theorem 6.1.1 — Invariance theorem. Let U and U' two universal Turing machines. Then, there exists a constant $C_{U,U'}$, that depends on U and U' , such that for each string $s \in \Sigma^*$ we have that:

$$K_U(s) \leq K_{U'}(s) + C_{U,U'}$$

Proof. Let p, v be the shortest strings that $U'(p, v) = s$, then we have that $U(\langle U', p, v \rangle, \lambda) = U'(p, v) = s$, and that $l(\langle U', p \rangle) = \log(l(\langle U' \rangle)) + 1 + l(\langle U' \rangle) + l(p) = K_{U'}(s) + C$, where $l(\langle U' \rangle)$ is the length of the machine U' using the encoding described in Section 4.2. ■

■ **Example 6.2** Consider a universal programming language, such as Java, and an alternative language, such as Python. We can write a Python interpreter in Java, that is, a Java program that takes a Python script as input and executes it. Then, to compute the complexity of a string $s \in \mathcal{B}^*$ using Java, $C_J(s)$, it would be no greater than the complexity of the string using Python, $C_P(s)$, plus the length of the Python interpreter written in Java, $C_{J,P}$. Importantly, the length of the interpreter, $C_{J,P}$, does not depend on the string s . ■

Although we have proved that Kolmogorov complexity does not depend on the selected universal Turing machine, the size of the constant $C_{U,U'}$ could pose a limitation in practical applications, especially when computing the complexity of short strings where the constant might significantly exceed the complexity of the string itself. This challenge is addressed by the Minimum Description Length principle, as described in Section 7.3.

the two parts definition $l(p) + l(v)$ because it is more in line with the requirements of the theory of nescience.

Notation 6.1. We denote by s^* the shortest program that outputs the string s on the universal Turing machine U , that is, $s^* = \langle p, v \rangle$, $U(s^*) = s$ and $l(s^*) = K(s)$. If more than one program satisfies these properties, we select the first one using a lexicographical order induced by $0 < 1$.

The size of the $C_{U,U'}$ constant is not the only challenge presented by Kolmogorov complexity; another issue is its non-computability, that is, there is no algorithm capable of determining the shortest program that generates any given string. The following theorem on the uncomputability of Kolmogorov complexity marks a pivotal insight into the intrinsic limits of complexity theory. By exploring the theorem, we delve into the heart of computability theory, confronting the paradoxical reality that some of the most fundamental aspects of information are inherently beyond the reach of algorithmic computation.

Theorem 6.1.2 The function $K : \mathcal{B}^* \rightarrow \mathbb{N}$ that assigns to each string s its Kolmogorov complexity $K(s)$ is not computable.

Proof. **TODO: Review** Assume for the sake of contradiction that K is computable. This means there exists a Turing machine T such that for any string s , T halts with $K(s)$ on its tape. Consider the following process, which uses T to construct a string s of arbitrary complexity: i) enumerate all binary strings using a shortlex ordering: s_1, s_2, \dots , ii) for each string s_i , compute $K(s_i)$ using T and select s_i such that $K(s_i) > l(s_i) + C$, where C is a constant representing the length of this description plus the operation to add C . Such a process would effectively produce a string s whose Kolmogorov complexity is greater than its length plus a constant, which contradicts the definition of Kolmogorov complexity as the length of the shortest description. The contradiction arises because our assumption that K is computable allows us to construct a string that defies the bounds set by K itself. Therefore, the function K is not computable. ■

If K were computable, we could solve the Halting Problem by constructing a program that, for any input program and input, computes whether the program halts by checking if its Kolmogorov complexity is finite. Since the Halting Problem is known to be undecidable, this provides a contradiction.

In practice, we approximate the Kolmogorov complexity using the compressed version of the string, employing standard compression algorithms, such as the Huffman algorithm described in Section 5.5.

6.2 Properties of Complexity

In this section, we delve into the properties of Kolmogorov complexity. We will explore the foundational principles that govern this complexity measure, including its invariance, symmetry, and non-computability. Through examining these properties, we gain deeper insights into the complex interplay between information, computation, and randomness.

Kolmogorov complexity is a finite positive natural number.

Proposition 6.2.1 For all $s \in \mathcal{B}^*$ we have that $0 < K(s) < \infty$.

Proof. Since $K(s)$ is the length of a string, it must be greater than 0. The property $K(s) < \infty$ is a consequence of Proposition 6.2.2 and the fact that we are only dealing with finite strings. ■

The Kolmogorov complexity of a string cannot surpass the sum of its own length and a constant.

Proposition 6.2.2 There is a constant c such that for all $s \in \mathcal{B}^*$ we have that $K(s) \leq l(s) + c$.

Proof. Let $s \in \mathcal{B}^*$ be an arbitrary string, and consider the encoding of a Turing machine p such that for any input $v = s$, it halts and outputs s . The program p is designed to simply reproduce its input.

Given this setup, when p is executed on a universal Turing machine U with s as input, it satisfies the condition $U(p, s) = s$. The length of p is a constant c across all strings s . This constancy arises because p 's mechanism (accepting an input and outputting it unchanged) does not vary with the size or content of s . By the definition of Kolmogorov Complexity $K(s)$, which seeks the minimum length of a program-input pair that generates s , the combination of p and s presents a feasible solution. Therefore, we have $K(s) \leq l(s) + l(p) = l(s) + c$. ■

The size of the constant c depends on the specific encoding schema used by the selected universal Turing machine U , but it is independent of the string s . In Section 6.6, we will explore the characteristics of random strings, which are defined as strings that cannot be compressed. Such strings exhibit a Kolmogorov complexity that exceeds their own length, that is, $K(s) > l(s)$.

The absolute difference in Kolmogorov complexity between any string x and its transformed counterpart $f(x)$, via a computable bijection, is bounded by a constant c . That is, not only does f not increase the complexity of x by more than a constant, but also f does not decrease the complexity by more than a constant.

Proposition 6.2.3 Let $f : \mathcal{B}^* \rightarrow \mathcal{B}^*$ be a computable bijection, then there exists a constant c such that $|K(f(x)) - K(x)| < c$.

Proof. Let P_f be the program that computes f and $P_{f^{-1}}$ the program that computes the inverse of f . For any string x , let P_x be the shortest program that generates x . Then, a program $P_{f(x)}$ that generates $f(x)$ can be constructed by concatenating P_x with P_f . The length of this program is $|P_{f(x)}| = |P_f| + |P_x|$. Since $|P_f|$ is a constant that does not depend on x , we can say that $K(f(x)) \leq K(x) + |P_f|$. Similarly, given $f(x)$, we can construct a program P'_x to generate x by applying $P_{f^{-1}}$ to $f(x)$. The length of this program is $|P'_x| = |P_f| + |P_{f^{-1}}|$. Thus, $K(x) \leq K(f(x)) + |P_{f^{-1}}|$. The two inequalities combined implies that $|K(f(x)) - K(x)| \leq \max(|P_f|, |P_{f^{-1}}|) = c$, where c is a constant that represents the maximum of the lengths of the programs that compute f and f^{-1} . This constant c does not depend on x , but rather on the complexity of the functions f and f^{-1} . ■

This proposition shows a remarkable stability of informational content under computable bijections, underscoring the intrinsic robustness of Kolmogorov complexity in the face of such transformations.

■ **Example 6.3** TODO: Provide an example clarifying this concept. ■

6.3 Joint Kolmogorov Complexity

The joint Kolmogorov complexity of two strings s and t is defined as the length of the shortest is the shortest program p that, when executed on an universal Turing machine U , outputs the pair $\langle s, t \rangle$, that is, in such a way that allows both strings to be unambiguously retrieved.

Definition 6.3.1 — Joint Kolmogorov Complexity. The *Joint Kolmogorov complexity* of the strings $s, t \in \mathcal{B}^*$, denoted by $K(s, t)$, is defined as:

$$K(s, t) = \min_{p, v \in \mathcal{B}^*} \{l(p) + l(v) : U(p, v) = \langle s, t \rangle\}$$

The notation $K(s, t)$ and $K(st)$ represent two different concepts in the context of Kolmogorov complexity. $K(s, t)$ refers to the joint Kolmogorov complexity of two strings s and t as per Definition 6.3.1, meanwhile $K(st)$ represents the Kolmogorov complexity of the concatenation of s and t , without any additional structure to distinguish between them, and so, Definition 6.1.2 is applied. The choice between $K(s, t)$ and $K(st)$ depends on whether it's important to preserve and utilize the distinction and relationship between s and t . If analyzing the interplay or the shared characteristics

of s and t is relevant, $K(s, t)$ is more appropriate. If the focus is on the information content of the combined sequence without regard to its origin from two separate strings, $K(st)$ is used.

■ **Example 6.4** TODO: Provide an example clarifying this concept. ■

Our first proposition highlights a fundamental symmetry in Kolmogorov complexity, illustrating that the complexity of describing a pair of strings in either order differs by at most a constant. This reflects the intrinsic property that the information content is independent of the specific arrangement of the strings being described. The constant c encapsulates the overhead associated with the operations needed to reverse the order of the strings.

Proposition 6.3.1 There is a constant c such that for all $x, y \in \mathcal{B}^*$ we have that $K(x, y) \leq K(y, x) + c$.

Proof. The key to this proof lies in the symmetry of information and the properties of a universal Turing machine. Let U be a universal Turing machine, and let p_{xy} and p_{yx} be the shortest programs that output x followed by y , and y followed by x , respectively, when executed on U . To convert p_{xy} into a program that outputs yx , we need to add or modify a finite set of instructions to swap the output order. This set of instructions has a fixed length, denoted by c . Therefore, the program that first computes x and y and then swaps their order can be at most c bits longer than the program that directly computes y and x . ■

Next proposition underscores the subadditive nature of Kolmogorov complexity, proving that the total complexity of describing two strings jointly cannot exceed the sum of their individual complexities by more than a fixed constant, irrespective of the strings' content.

Proposition 6.3.2 There is a constant c such that for all $s, t \in \mathcal{B}^*$ we have that $K(s, t) \leq K(s) + K(t) + c$.

Proof. Let s^* and t^* be the shortest computer programs that generate s and t , respectively. Given that s^* and t^* are designed to be prefix-free, a universal Turing machine U can unambiguously decode each program without needing any additional information and subsequently generate the strings s and t . The constant c accounts for the requirement of U to prepend the concatenated program for st with the length $l(s)$ of s . This step ensures that the concatenated sequence st can be decoded unambiguously. ■

The last proposition states the relationship between the complexity of individual strings and their joint complexity, by establishing a lower bound on the joint complexity of two strings relative to their individual complexities.

Proposition 6.3.3 There is a constant c such that for all $s, t \in \mathcal{B}^*$ we have that $K(s, t) \geq K(s) + c$ and $K(s, t) \geq K(t) + c$.

Proof. Assume for the sake of contradiction that $K(s, t) < K(s) + c$. Let p be the shortest computer program that outputs the pair $\langle s, t \rangle$, meaning that $K(s, t) = l(p)$. Since program p unambiguously generates s as part of the output pair, it follows that the complexity of generating s alone, $K(s)$, should not exceed the length of p , which is a contradiction with respect to the initial assumption of $K(s, t) < K(s) + c$. ■

6.4 Conditional Kolmogorov complexity

In this section, we explore the concept of *conditional Kolmogorov complexity*, which examines how the complexity of a string s can be significantly decreased by assuming prior knowledge of another string t . This notion highlights the impact of assuming some background information on the compressibility of a description.

Definition 6.4.1 — Conditional Kolmogorov Complexity. The *conditional Kolmogorov complexity* of a string $s \in \mathcal{B}^*$ given the string $t \in \mathcal{B}^*$ is defined as:

$$K(s|t) = \min_{p,v \in \mathcal{B}^*} \{l(p) + l(v) : U(p, \langle v, t \rangle) = s\}$$

Similar to the non-conditional Kolmogorov complexity, the conditional complexity of a string s , given a background string t , depends on the strings themselves rather than the specific universal Turing machine used for computation. That is, for any two universal Turing machines, U and U' , it is easy to show that there exists a constant $C_{U,U'}$ such that for all strings $s, t \in \Sigma^*$, the inequality $K_U(s|t) \leq K_{U'}(s|t) + C_{U,U'}$ holds. This property underscores the inherent machine-independence of conditional complexity measures.

■ **Example 6.5 TODO:** Provide an example clarifying this concept. ■

As it was the case of the unconditional Kolmogorov complexity, conditional Kolmogorov complexity is a finite positive natural number.

Proposition 6.4.1 For all $s, t \in \mathcal{B}^*$ we have that $0 < K(s|t) < \infty$.

Proof. Since $K(s)$ is the length of a string, it must be greater than 0. The property $K(s|t) < \infty$ is a consequence of Proposition 6.2.2 and the fact that we are only dealing with finite strings. ■

Next proposition posits that when a string s is conditioned upon itself, its complexity stabilizes to a universal constant c .

Proposition 6.4.2 There is a constant c such that for all $s \in \mathcal{B}^*$ we have that $K(s|s) = c$.

Proof. When a string s is conditioned on itself, the information needed to generate s from s can be encapsulated in a Turing machine that simply copies its input to its output. This machine, being independent of the specific content of s , has a fixed length c . ■

This proposition explores the relationship between unconditional and conditional Kolmogorov complexities, establishing an upper bound for the latter. It asserts that for any strings s and t , the complexity of s given t is at most the complexity of s alone, plus a constant c . This highlights the intuitive notion that having additional information at most reduces the complexity of describing a string, or in the worst case, adds a constant overhead, but does not increase it beyond this bound.

Proposition 6.4.3 There is a constant c such that for all $s, t \in \mathcal{B}^*$ we have that $K(s|t) \leq K(s) + c$.

Proof. There exists a Turing machine T' that ignores its input and t directly generates s . The length of T' is $K(s)$ by definition. ■

The relationship between conditional, unconditional, and joint Kolmogorov complexities, offers a comprehensive perspective on the informational interdependencies of binary strings. It posits that the complexity of a string s given another string t is at most the complexity of s alone, which in turn is no greater than the joint complexity of both s and t . The proposition encapsulates the intuitive understanding that knowledge of additional data (in this case, t) cannot increase the complexity of describing s , and that the combined description of two strings is at least as complex as describing each independently.

Proposition 6.4.4 For all $s, t \in \mathcal{B}^*$ we have that $K(s|t) \leq K(s) \leq K(s, t)$.

Proof. Given Proposition 6.4.3 and Proposition 6.3.3. ■

The Kolmogorov complexity chain rule is a fundamental principle that connects the joint complexity of two strings with their individual and conditional complexities. It asserts that the total complexity of a pair of strings s and t can be decomposed into the complexity of s plus the complexity of t given s . This relationship mirrors the additive property of entropy in information theory (see Proposition XXX) and provides a powerful tool for understanding the interplay between information content and conditional information in the context of Kolmogorov complexity.

Proposition 6.4.5 For all $s, t \in \mathcal{B}^*$ we have that $K(s, t) = K(s) + K(t | s)$

Proof. TODO ■

■ **Example 6.6** TODO: Provide an example clarifying this concept. ■

6.5 Information Distance

In this section, we aim to introduce a universal metric for quantifying the absolute information distance between two or more individual entities encoded as strings of symbols. Intuitively, the information distance between two strings s and t can be understood as the length of the shortest computer program for a universal computer that enables the generation of s given t and vice versa.

Definition 6.5.1 The *information distance* between two strings $s, t \in \mathcal{B}^*$ with respect to a universal Turing machine U , denoted by $ID_U(s, t)$, is defined as

$$ID_U(s, t) = \min\{l(p) : U(p, s) = t, U(p, t) = s\}$$

It can be shown that for any pair of universal Turing machines, U_1 and U_2 , the information distance between two strings s and t differs by no more than a constant c ; this constant c depends on the choice of U_1 and U_2 but is independent of the specific strings s and t . It's also important to note that despite its theoretical utility, information distance is non-computable, meaning it does not exist and algorithm to calculate it in practice.

The idea of using the conditional Kolmogorov complexity of s given t , that is $K(s | t)$, as a metric for information distance may seem appealing. However, this approach is unsuitable for capturing the essence of information distance because of its inherent asymmetry (refer to Proposition XXX). For example, the complexity $K(\lambda | t)$ remains minimal even when t significantly differs from the empty string. Alternatively, employing the sum $K(s | t) + K(t | s)$ as a measure of distance is also inadequate, as it overlooks the redundancy in the information necessary for transforming s into t and vice versa.

Next proposition shows the proper way in which the information distance between two strings can be computed using their conditional Kolmogorov complexity.

Proposition 6.5.1 Let $s, t \in \mathcal{B}^*$ be two binary strings, then we have that:

$$ID_U(s, t) = \max\{K(s | t), K(t | s)\} + O(\log \max\{K(s | t), K(t | s)\})$$

Proof. TODO: based on the maximal overlap. ■

■ **Example 6.7** TODO: Give an example based on the bitwise exclusive or. ■

Introduce the concept of max distance as:

$$E(x, y) = \max\{K(x | y), K(y | x)\}$$

Introduce the following proposition.

Proposition 6.5.2 $E(x, y)$ is a metric.

Proof.

Introduce the following proposition.

Proposition 6.5.3

$$E(x,y) = \max\{K(x|y), K(y|x)\} = K(xy) - \min\{K(x), K(y)\} + O(\log K(xy))$$

Proof. TODO: Pending

TODO: Introduce the concept of admissible information distance. Prove that $E(x,y)$ is an admissible information distance. Prove that it is minimal for every admissible information distance. Explain this notion of universality.

This concept of information distance is universal because it encompasses all other computable distance metrics as special cases.

Normalized Information Distance

Information distance is an absolute measure; however, when assessing similarity, we are often more concerned with relative measures. For instance, consider two strings each of length 1,000,000 that differ by 1000 bits; we would perceive these strings as being relatively more similar compared to two strings of length 1000 bits that differ by the same amount of 1000 bits. This concept of normalization implies that the size of the description required for transformation should be evaluated in the context of the sizes of the objects involved.

Definition 6.5.2 The *normalized information distance* between two binary strings $s, t \in \mathcal{B}^*$, denoted by $e(s, yt)$, is defined as:

$$NID(s,t) = \frac{\max\{K(s|t), K(t|s)\}}{\max\{K(s), K(t)\}}$$

As expected, the normalized information distance is a number between zero and one.

Proposition 6.5.4 The normalized information distance $NID(s,t)$ takes values in the range $[0, 1]$.

Proof. TODO: Pending

Introduce the following proposition.

Proposition 6.5.5 The normalized information distance $NID(x,y)$ is a metric, up to negligible errors.

Proof. TODO: Pending

Introduce the following proposition.

Proposition 6.5.6

$$NID(x,y) = \frac{K(xy) - \min\{K(x), K(y)\} + O(\log K(xy))}{\max\{K(s), K(t)\}}$$

Proof. TODO: Pending

Normalized Compression Distance

Although the normalized information distance metric is not computable, it has a potential wide range of applications. By approximating K with practical compressors, where $Z(s)$ represents the binary length of the string s compressed using a compressor Z (such as "gzip", "bzip2", "PPMZ"), we can approximate the concept of normalized information distance.

Definition 6.5.3 The *normalized compression distance* between two strings $s, t \in \mathcal{B}^*$, given the compressor Z , and denoted by $NCD_Z(s, t)$, is defined as:

$$NCD_Z(s, t) = \frac{\max\{Z(s | t), Z(t | s)\}}{\max\{Z(s), Z(t)\}}$$

Unfortunately, for the majority of the compressors it is very difficult to compute the compressed conditional version of a string $Z(s | t)$. Fortunately, we can rewrite the NCD in a different way in which we do not need to use conditional compressions.

Proposition 6.5.7 The normalized compression distance between two strings $s, t \in \mathcal{B}^*$, given the compressor Z , satisfies:

$$NCD_Z(s, t) = \frac{Z(st) - \min\{Z(s), Z(t)\}}{\max\{Z(s), Z(t)\}}$$

Proof. TODO: prove. ■

The normalized compression distance constitutes a spectrum of distances, each defined by the choice of compressor Z . The effectiveness of Z determines how closely the normalized compression distance mirrors the normalized information distance, ultimately influencing the quality of the outcomes.

6.6 Incompressibility and Randomness

TODO: Section Pending!

It is easy to prove that the majority of strings cannot be compressed. For each n there are r^n strings of length n , but only $\sum r^k$ possible shorter descriptions. Therefore, there is at least one string x of length n such that $K(n) \geq n$. We call such strings *incompressible*.

Definition 6.6.1 For each constant c we say that a string s is *c-incompressible* if $K(s) \geq l(s) - c$.

We will use the term *incompressible string* to refer to all the strings that are c -incompressible with small c . Those strings are characterized because they do not contain any patterns, since a pattern could be used to reduce the description length. Intuitively, we think of such patternless sequences as being *random*.

The number of strings of length n that are c -incompressible is at least $2^n - 2^{(n-c)} + 1$. Hence there is at least one 0 -incompressible string of length n , at least one-half of all strings of length n are 1 -incompressible, at least three-fourths of all strings of length n are 2 -incompressible, ..., and at least the $(1 - 1/2^c)$ th part of all 2^n strings of length n are c -incompressible. This means that for each constant $c > 1$ the majority of all strings of length n with $n > c$ are c -incompressible.

Bibliography and References

TODO: Section Pending!

Kolmogorov Complexity, a fundamental concept in the field of information theory and computer science, emerged in the 1960s from the work of three pioneering researchers: Andrei Kolmogorov, Ray Solomonoff, and Gregory Chaitin. Independently arriving at similar concepts, these scholars

sought to formalize a measure of the information content or complexity of an object (specifically, finite strings) in a way that is independent of the specific language or encoding used. Kolmogorov, a prominent Soviet mathematician, introduced his formulation in 1965, aiming to provide a theoretical foundation for understanding the complexity of individual objects beyond the probabilistic framework of Shannon's information theory. Solomonoff, an American information theorist, introduced a related concept as part of his work on algorithmic probability, emphasizing its implications for prediction and inductive inference. Chaitin, working in the United States as well, contributed by exploring the mathematical properties and implications of this notion, including its non-computability. The motivations behind the development of Kolmogorov Complexity were multifaceted, encompassing the desire to better understand the nature of information, randomness, and the limits of computation and prediction, thus laying the groundwork for numerous applications in theoretical computer science, mathematics, and beyond.

The idea of measuring the amount of information contained in a string based on the shorter computer program that can reproduce it was introduced independently by Solomonoff [[solomonoff1964formal](#)], Kolmogorov [[kolmogorov1965three](#)] and Chaitin [[chaitin1969simplicity](#)].

In the Kolomogorov complexity literature, the concept of Kolmogorov complexity is defined in terms of computable functions, i.e. Turing machines. Then it is shown that there are some machines that do not provide worse descriptions than others, up to a constant. And finally, it is proved that universal Turing machines are not worse than any other machines. We have provided here an easier to understand short-cut to this approach. Also, we do not consider the case of of non-prefix Kolmogorov complexity. And we prefer to split code and data. See XX, XX, XX, or XX for a more classical introduction to Kolmogorov complexity.



7. Learning

*Some mathematical statements are true for no reason,
they're true by accident.*

Gregory Chaitin

Warning: This section still requires a significant amount of work!

Machine learning refers to a large collection of algorithms designed to automatically build mathematical models based on sample data sets, usually with the aim of making predictions, classifying objects, or simply to better understand the structure of the data. In the past decade, machine learning algorithms have been highly successful in areas like self-driving cars, practical speech recognition, effective web search, and purchase recommendations.

In this chapter we are going to see how the problem of learning from data is formally formulated in the area of machine learning. In this sense, the chapter is a continuation of the introduction to discrete probability included in Section ???. Also, we are going to study in detail two particular approaches to machine learning that are highly related to our theory of nescience: the Minimum Description Length principle and the Minimum Message Length principle.

Most of the learning algorithms used today in practice are known since forty years ago. The high success of current machine learning applications is largely due to the availability of huge, high-quality, training datasets, and to the advance of computing power, and in particular, thanks to the powerful graphical processing units (GPU) used in video-games. In Chapter 17 we will introduce a collection of new machine learning algorithms based on the theory of nescience, and we will compare them with the current, state of the art, algorithms.

7.1 Statistical Inference

Move to the right place. A *parametric random variable* X , denoted by $f(X | \theta)$, is a distribution that belongs to a family of functions parameterized by θ , where θ can be a single parameter, or a vector of several parameters.

Statistical inference is the branch of probability and statistics concerned with deducing the probabilistic model behind a population after analyzing some data that, we believe, contain relevant

information. Statistical inference is different from the area of *descriptive statistics*, where the goal is to provide a summary of the main properties of the observed data, and *probability theory*, that deals with the theoretical foundations. From the area of statistical inference, we are mostly interested in *model selection* and *point estimation*. Other techniques, like *confidence intervals*, *hypothesis testing* or *experiment design* are not covered in this short review since they are not needed in the book.

Definition 7.1.1 A *statistical model* is a random variable, together with a specification of its probability distribution, and the identification of the parameters, denoted by θ , of that distribution. When the parameter θ is unknown, it is said that the distribution of the random variable is conditional to θ .

We assume that the actual value of the unknown parameter θ can be inferred, usually through a collection of data samples. The parameter θ could be a single scalar or a vector of values, and it is also considered a random variable that follows a particular probability distribution. The set $\Theta = \{\theta_1, \theta_2, \dots\}$ composed by all the possible values of the parameter θ is called the *parameter space*.

■ **Example 7.1** *TODO: Rewrite this example, or find another one.* The binomial distribution with parameters n and p is a model for a family of experiments in which we are interested to know the number of successes in a sequence of n independent binary trials (that is, each trial could be either a success or a failure), being the probability of success p . If X is a random variable following a binomial distribution, denoted by $X \sim B(n, p)$, the probability of getting exactly k successes is given by:

$$Pr(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

In statistical inference we are usually interested in the inverse problem. That is, we have the actual result of an experiment composed by n samples, in which we know how many success k we have got, and we would like to know the probability p of success. ■

On the contrary of what happens with logical deduction, where we can be sure about the truthiness of a conclusion, in case of induction and inference, the knowledge gathered is not conclusive. Probability theory is the tool we use to deal with the uncertainty of statistical inferences (probabilistic statements about one of the elements of a statistical model).

Definition 7.1.2 Let X_1, \dots, X_n be n random variables. A *statistic* is a random variable $T = r(X_1, \dots, X_n)$, where $r()$ an arbitrary real-valued function of n variables.

The sample mean and the sample variance are examples of statistics. Statistics allow us to identify the elements of the inference in which we are interested, and to evaluate quantitatively the quality of the results.

Definition 7.1.3 Let f be the distribution of a random variable X with parameter θ . The probability distribution of the parameter θ , denoted by $\xi(\theta)$, is called the *prior distribution*.

The prior distribution is also the marginal distribution of $f(x | \theta)$ for all the possible samples x . The prior distribution must be defined over the parameter space Ω . $\xi(\theta)$ is called the prior distribution because it is the distribution of the parameter θ that we have before observing any samples of data. Normally, the prior distribution is derived based on theoretical considerations about our current knowledge of the experiment, and so, they might be incorrect.

Definition 7.1.4 Let f be the distribution of a random variable X with parameter θ , and let X_1, \dots, X_n be n random variables. The conditional probability distribution of the parameter θ

given the observed values $X_1 = x_1, \dots, X_n = x_n$, denoted $f(\theta | x_1, \dots, x_n)$, is called the *posterior distribution*.

Baye's theorem is the tool we have at our disposal to derive the posterior distribution given a prior distribution and the observations.

Proposition 7.1.1 Suppose that the n random variables X_1, \dots, X_n form a random sample from a distribution for which the p.d.f. or de p.f. is $f(x | \theta)$. Suppose also that the value of the parameter θ is unknown and the prior p.d.f. or p.f. of θ is $\psi(\theta)$. Then the posterior p.d.f. or p.f. of θ is

$$\xi(\theta | \mathbf{x}) = \frac{f(x_1 | \theta) \cdots f(x_n | \theta) \xi(\theta)}{g_n(\mathbf{x})} \quad \text{for } \theta \in \Omega$$

where g_n is the marginal distribution of X_1, \dots, X_n

$$g_n(\mathbf{x}) = \int_{\Omega} f_n(\mathbf{x} | \theta) \xi(\theta) d\theta$$

The consequences of assuming a wrong prior distribution can be mitigated by increasing the number of observations.

■ Example 7.2

If they are required in the text, extend this section with the following topics: How to estimate priors (maximum likelihood, uninformative, and maximum entropy). Conjugate priors. Sensitivity analysis.

Log-likelihood [...] natural logarithm is a monotonically increasing function [...] ensures that the maximum value of the log of the probability occurs at the same point as the original probability function [...]

An estimator of a parameter is some function of the data that we hope is close to the parameter.

Definition 7.1.5 Let X_1, \dots, X_n be observable data whose joint distribution is indexed by a parameter θ taking values in a subset Ω of the real line. An *estimator* of the parameter θ is a real-valued function $\delta(X_1, \dots, X_n)$. If $X_1 = x_1, \dots, X_n = x_n$ are observed, then $\delta(X_1, \dots, X_n)$ is called the *estimate* of θ .

TODO: Relate the concepts of statistic and estimator. Provide a more generic definition of estimator.

The estimator itself is a random variable, and its probability distribution can be derived from the joint distribution of X_1, \dots, X_n if desired.

Definition 7.1.6 A *loss function* is a real-valued function of two variables, $L(\theta, a)$, where $\theta \in \Omega$ and a is a real number. The interpretation is that the statistician loses $L(\theta, a)$ if the parameter equals θ and the estimate equals a .

TODO: How this definition relates to other definitions of loss functions?

Let $\psi(\theta)$ denote the prior p.d.f. of θ on the set Ω . If the statistician chooses a particular estimate a , then her expected loss will be

$$E[L(\theta, a)] = \int_{\Omega} L(\theta, a) \xi(\theta) d\theta$$

The statistician wishes to choose an estimate a for which the expected loss is a minimum.

7.1.1 Bayesian Inference

In Bayesian statistical inference we assume that probabilistic knowledge about the data source is available before collecting observations, what it is called *prior probability*. Let Θ be a family

of models with a discrete parameter $\{\theta_1, \theta_2, \dots\}$ and with prior probabilities $Pr(\theta_i)$. θ_i could be a single scalar value, or a vector value. We also assume that the likelihood $Pr(x | \theta_i)$ is known. Applying Bayes' theorem we have that the posterior probability $Pr(\theta_i | x)$ is given by

$$Pr(\theta_i | x) = \frac{Pr(x | \theta_i) Pr(\theta_i)}{Pr(x)} \quad \forall \theta_i \in \Theta$$

where $Pr(x)$ is the marginal data probability $Pr(x) = \sum_{\theta_i} Pr(x | \theta_i) Pr(\theta_i)$. The value $Pr(x)$ is just a normalizing constant to be sure that $Pr(\theta_i | x)$ is a probability between 0 and 1. We usually remove this value and make the following approximation $Pr(\theta_i | x) \propto Pr(x | \theta_i) Pr(\theta_i)$. If we need a single estimated value for θ , we use the mode of $Pr(\theta)$, what it is called the Maximum a Posteriori, or MAP, estimation. When our prior distribution $Pr(\theta_i)$ is the uniform distribution, MLE and MAP provide the same result.

A Bayes estimator is an estimator that is chosen to minimize the posterior mean of some measure of how far the estimator is from the parameters, such as squared error or absolute error.

Suppose we can observe the value \mathbf{x} of the random vector \mathbf{X} before estimating θ , and let $\xi(\theta | \mathbf{x})$ denote the posterior p.d.f. of θ on Ω . For each estimate a the expected loss will be

$$E[L(\theta, a) | \mathbf{x}] = \int_{\Omega} L(\theta, a) \xi(\theta | \mathbf{x}) d\theta$$

Definition 7.1.7 Let $L(\theta, a)$ be a loss function. For each possible value \mathbf{x} of \mathbf{X} , let $\delta^*(\mathbf{x})$ be a value of a such that $E[L(\theta, a) | \mathbf{x}]$ is minimized. Then δ^* is called a *Bayes estimator* of θ . Once $\mathbf{X} = \mathbf{x}$ is observed, $\delta^*(\mathbf{x})$ is called a *Bayes estimate* of θ .

Another way to describe a Bayes estimator δ^* is to note that, for each possible value \mathbf{x} of \mathbf{X} , the value $\delta^*(\mathbf{x})$ is chosen so that

$$E[L(\theta, \delta^*(\mathbf{x})) | \mathbf{x}] = \min_a E[L(\theta, a) | \mathbf{x}]$$

The Bayes estimator will depend on both the loss function that is used in the problem and the prior distribution that is assigned to θ .

■ **Example 7.3** The loss function $L(\theta, a) = (\theta - a)^2$ is called squared error loss. Let θ be a real-valued parameter. Suppose that the squared error loss function is used and that the posterior mean of θ , $E(\theta | \mathbf{X})$, is finite. Then, a Bayes estimator of θ is $\delta^*(\mathbf{X}) = E(\theta | \mathbf{X})$.

The loss function $L(\theta, a) = |\theta - a|$ is called absolute error loss. When the absolute error loss function is used, a Bayes estimator of a real-valued parameter $\delta^*(\mathbf{X})$ equal to a median of the posterior distribution of θ . ■

Definition 7.1.8 A sequence of estimators that converges in probability to the unknown value of the parameter being estimated, as $n \rightarrow \infty$, is called a consistent sequence of estimators.

Under fairly general conditions and for a wide class of loss functions, the Bayes estimators of some parameters θ will form a consistent sequence of estimators as the sample size $n \rightarrow \infty$.

The theory of Bayes estimators provides a satisfactory and coherent theory for the estimation of parameters. To apply the theory, it is necessary to specify a particular loss function, and also a prior distribution for the parameter. It is specially difficult to specify a meaningful prior distribution on the multidimensional parameter space Ω .

7.1.2 Non-Bayesian Inference

We would like to infer a statement about which model, or which parameters for a model, we should prefer given the collected data.

Definition 7.1.9 Let f be the distribution of a random variable X with parameter θ . We call $Pr(X | \theta)$ the *likelihood function*.

Maximum likelihood estimation is a method that determines values for the parameters of a model. The parameter values are found such that they maximise the likelihood that the process described by the model produced the data that were actually observed.

Let Θ be a family of models (probability distributions) with a discrete parameter $\{\theta_1, \theta_2, \dots\}$ and with prior probabilities $Pr(\theta_i)$, θ_i could be a single scalar value or a vector value, and let x the data collected iid. For each model θ , we assume the probability of getting data x given model θ , $Pr(x | \theta)$, is known. The *Maximum Likelihood Estimator*, or MLE, selects as best estimate the value of θ that maximizes the likelihood $f(x | \theta)$.

7.2 Machine Learning

Rewrite this section using a schema of definition-proposition-proof.

There is a controversy between mathematicians and computer scientists about the difference between statistical inference and machine learning. From our point of view there are important differences. In statistical inference, given a target variable y and a training data X , the goal is to find a model that infer a value \hat{y}_i that maximizes the probability $P(\hat{y}_i | x_i)$; meanwhile, in machine learning, finding the value with the highest probability is not necessarily our objective, perhaps we are more interested in minimize the number of false positives, or minimize the number of errors among under-represented categories (see below). Another difference is that statistical inference models are mathematical functions, meanwhile machine learning models are usually computer programs that can not be easily manipulated algebraically. Finally, in statistics we are also interested in the mathematical properties of the methods in use, that is, if there exists a solution, if we can guarantee that search algorithms converge, and to estimate how far the selected models are from the real ones; in machine learning our interest is mostly finding models that work in practice, without caring too much about their mathematical properties. The goal of this book is to reconcile both worlds into one single theoretical framework.

Broadly speaking, machine learning algorithms can be classified into two main categories: supervised and unsupervised. In *supervised* learning we have a collection of training samples and the corresponding observed target values, and our interests is to predict the output of new, previously unseen, observations; meanwhile in *unsupervised* learning there are no targets, just training samples, and what we are looking for is to learn the structure of the data. Supervised learning algorithms can be applied to *regression* problems, where the value to predict is quantitative, and to solve *classification* problems, where the targets are qualitative. *Qualitative* attributes take values from a finite collection of distinct non-numerical categories, and so, no arithmetic operations can be applied to them, although in some cases they can be ranked in order. *Quantitative* attributes are numerical, and they can be either *discrete* if the range of possible values is countable, or *continuous* if it is not countable.

Let $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$ be a training dataset composed by p features, where each individual feature $\mathbf{x}_i = \{x_{i1}, x_{i2}, \dots, x_{in}\}$ is composed by n observed values, and let $\mathbf{y} = \{y_1, \dots, y_n\}$ be a target variable. We assume there is some relationship between the target values y_i and the samples \mathbf{x}_i that can be expressed in the form

$$\mathbf{y} = f(\mathbf{X}) + \epsilon \tag{7.1}$$

where f is a unknown function, and ϵ is a random terms independent of \mathbf{X} and with mean zero. Our goal is to find a function \hat{f} , estimated using a machine learning algorithm, such that $\mathbf{y} \approx \hat{f}(\mathbf{X})$.

This function \hat{f} allows us to *predict* the target value, denoted by \hat{y} , for predictors not contained in the training dataset $\mathbf{x} \notin \mathbf{X}$

$$\hat{y} = \hat{f}(\mathbf{x})$$

Most of the statistical learning methods can be characterized as either parametric or non-parametric. With parametric methods we select a priori a functional form, and we fit the free parameters of this functional form. In contrast, with non-parametric models we do not make any assumption about the form of the models. Given their flexibility, non-parametric methods usually require more training data to train. Moreover, the risk of overfitting training data is in general higher in case of non-parametric methods than in case of parametric methods. Non-parametric methods are more difficult to interpret by humans.

There are two main reasons why we want to estimate the function f : prediction and inference. In case of inference, we are interested in learning the way that the response variable Y is affected when the predictors X change, but not necessarily to make predictions of future values. Depending on whether we are interested in prediction or inference, different machine learning methods are usually used.

7.2.1 Model Accuracy

The error term ε introduced in Equation 7.1 corresponds to variables that have not been taken into account in our study, and other effects that cannot be measured. This kind of error is called *irreducible*, since there is nothing we can do to reduce it. A second type of error, called *reducible*, refers to the fact that our estimate \hat{f} of the function f might not be perfect. It is called reducible because with better estimates the error will decrease.

Derive this property

We are interested in the average error made by our model. Assuming that both \hat{f} and X are fixed, it can be shown that

$$E(Y - \hat{Y})^2 = E[f(X) + \varepsilon - \hat{f}(X)]^2 = [f(X) - \hat{f}(X)]^2 + Var(\varepsilon)$$

The irreducible error is an upper bound to the accuracy of our predictions. Unfortunately, in practice, this bound is almost always unknown.

Mean Squared Error

A common metric used to quantitatively evaluate and compare the performance of the different machine learning algorithms is to compute the mean square error (MSE) of the predictions made,

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{f}(X_i))^2$$

where Y_i is the i th observed value, and the $\hat{f}(X_i)$ is the prediction that \hat{f} gives for the i th vector of predictors.

Explain how MSR relates to MLE

We are interested in the capability of the model \hat{f} to generalize to previously unseen data, that is, to correctly make predictions based on input vectors not included in the training dataset \mathcal{X} . In this sense, our goal should be to select that method with the lowest MSE over a test dataset, that is, over a collection of input vectors that have not been used for the training of the algorithm. When a model \hat{f} has a very low training MSE but a very high test MSE we say that the model overfits the training data.

If the response variable is qualitative the quantity we seek to minimize is the average number of misclassification made by the model, that is,

$$\frac{1}{n} \sum_{i=1}^n I(Y_i \neq \hat{f}(X_i))$$

where $\hat{f}(X_i)$ is the predicted class for the i th observation, and I is a function that equals 1 if $Y_i \neq \hat{f}(X_i)$ and zero otherwise.

7.2.2 No free lunch theorem

There is no free lunch in statistics: no one method dominates all others over all possible data sets. On a particular data set, one specific method may work best, but some other method may work better on a similar but different data set

7.2.3 The bias-variance trade-off

It is possible to show that the expected test MSE, for a given value x_0 , can be always decomposed into the sum of three fundamental quantities: the variance of $\hat{f}(x_0)$, the squared bias of $\hat{f}(x_0)$ and the variance of the error term ε :

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) [Bias(\hat{f}(x_0))]^2 + Var(\varepsilon)$$

where $E(y_0 - \hat{f}(x_0))^2$ defines the expected test MSE, and refers to the average test MSE that would obtain if we repeatedly estimated f using a large number of training sets, and tested each at x_0 .

In order to minimize the expected test error, we need to select a statistical learning method that simultaneously achieves low variance and low bias. The expected test MSE can never lie below $Var(\varepsilon)$, the irreducible error. Variance refers to the amount by which \hat{f} would change if we estimated it using a different training data set. In general, more flexible statistical methods have higher variance. Bias refers to the error that is introduced by approximating a real-life problem, which may be extremely complicated, by a much more simpler model. Generally, more flexible methods result in less bias. As a general rule, as we use more flexible methods, the variance will increase and the bias will decrease. The relative rate of change of these two quantities determines whether the test MSE increases or decreases.

The relationship between bias, variance, and test set MSE is referred to as the bias-variance trade-off. It is easy to obtain a method with extremely low bias but high variance, or a method with very low variance but high bias. The challenge lies in finding a method for which both the variance and the squared bias are low. In a real-life situation in which f is unobserved, it is generally not possible to explicitly compute the test MSE, bias, or variance for a statistical learning methods. Alternative approaches, like for example cross-validation, are used to estimate the test MSE using the training data.

7.2.4 Generative vs. discriminative models

TODO: Pending

7.2.5 Decision Trees

A decision tree is a mathematical model f that predicts the value of a target variable y by learning simple if-else decision rules inferred from the training set (\mathbf{X}, y) (see Example 7.4). Trees are simple and easy to interpret, but they do not give good accuracy.

The nodes of the tree contain pairs of values (j, w) , where $1 \leq j \leq p$ is a feature index and $w \in \mathbb{R}$ is a threshold, and the tree leafs contain labels of \mathcal{G} in case of a classification problem,

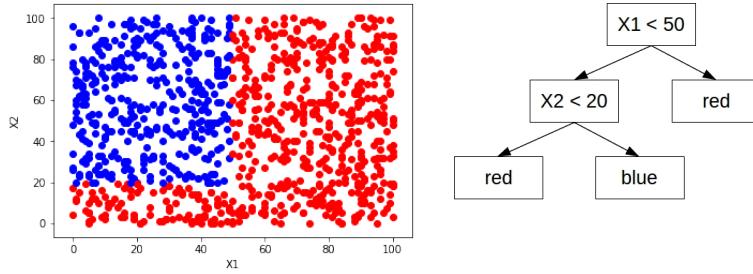


Table 7.1: Example of Decision Tree

or numbers in case of a regression problem. Given a vector $\mathbf{x} \in \mathbb{R}^p$ we perform a tree traversal checking at each node if $x_j \leq w$ to decide if we continue with the left or right branch of the node, until a leaf is reached. We associate the value \hat{y} of the reached leaf with the vector \mathbf{x} .

■ **Example 7.4** In Figure 7.1, left side, it is shown an example of a dataset composed by two classes, red dots and blue dots. We want to find a decision tree such that given the features X_1 and X_2 , it returns if the corresponding dot is blue or red. A possible solution to this problem is depicted in the right side of the figure. This decision tree can be also encoded as a function in a programming language, for example in Python, as next code shows.

```

def tree(X1, X2):
    if X1 < 50:
        if X2 < 20:
            return "red"
        else:
            return "blue"
    else:
        return "red"
  
```

The algorithms for the construction of decision trees usually work by recursively partitioning the training set \mathbf{X} in such a way that the values of the target vector \mathbf{y} are grouped together, until all partitions are composed by a single label. The problem with these building methods is that they produce very complex trees that overfit the training data. Overfitted trees not only lead to poor predictive capabilities on non-training data, but also produce models that can be exceedingly difficult to interpret. A common approach to avoid overfitting in decision trees is to force an early stopping of the algorithm before the tree becomes too complex. Popular stopping criteria include limiting the maximum depth of the tree, requiring a minimum number of sample points at leaf nodes, or computing the accuracy gain yielded by adding new nodes. However, those heuristics demand the optimization of hyperparameters which makes the training process computationally expensive.

TODO: briefly describe bagging, random forests, and boosting [...] produce multiple trees which are then combined to yield a single consensus prediction [...] combining a large number of trees can often result in dramatic improvement in prediction accuracy, at the expense of some loss of interpretability [...]

7.2.6 Time Series Analysis

A time series is a sequence of measurements taken at successively equally spaced points in time, so there exists a natural ordering of the observations. Examples of time series include the daily closing prices of the Standard and Poor's 500 index, the monthly number of passengers of an airline, or the yearly gross domestic product of a country.

Definition 7.2.1 A *time series* of length $n \in \mathbb{N}$, denoted by $\{x_t : t = 1, \dots, n\}$ or $\{x_t\}$, is a sequence $\{x_1, x_2, \dots, x_n\}$ of *observed values*.

The elements x_i of the series correspond to values sampled at fixed time intervals $1, 2, \dots, n$. The sampling interval must be short enough to provide a very close approximation to the original continuous signal. Observed values could be continuous, discrete or even categorical.

In statistics, a time series is usually represented as a sequence of n random variables, and a particular time series is a realization of this representation. In this sense, $\{x_t\}$ would denote a collection of random variables. In this book, we do not follow this approach for time series formalization.

Time series forecasting refers to the process of building a model to predict future values of the series based on the previously observed values. Time series forecasting is based on identifying the mean features in data and the random variation, and it is generally based on the assumption that present characteristics will continue in the near future, something that cannot be validated in practice.

Notation 7.1. Given the time time series $\{x_t : t = 1, \dots, n\}$ we denote $\hat{x}_{t+k|t}$ the forecast made at time t for a future value at time $t + k$, where k is the number of steps in the future.

Trends and Seasons

Many time series ... and a repeating seasonal component.

a systematic change in a time series that does not appear to be periodic is known as a trend. [...] A repeating pattern [...] within any fixed period [...] is known as seasonal variation [...] cycles [...] do not correspond to some fixed natural period. [...]

trend [...] change direction in unpredictable times [...] stochastic trend [...]

The main features of many time series are trends and seasonal variations that can be modelled deterministically with mathematical functions of time.

it is usually appropriate to remove trends and seasonal effects before comparing multiple series.

Definition 7.2.2 Let $\{x_t\}$ be a time series, a *simple additive model* is defined as

$$x_t = m_t + s_t + z_t$$

where m_t is called the *trend component*, s_t is the *seasonal component*, and z_t is the *error term*.

If the time series presents the property that the seasonal component increases as the trend increases, it might be better to use a multiplicative model.

Definition 7.2.3 Let $\{x_t\}$ be a time series, a *simple multiplicative model* is defined as

$$x_t = m_t s_t + z_t$$

where m_t is called the *trend component*, s_t is the *seasonal component*, and z_t is the *error term*.

In practice, a simple approach of estimating the trend of a time series is to compute a moving average.

Definition 7.2.4 Let $\{x_t\}$ be a time series, a *simple moving average* of length l is

The best results are achieved when the length l of the moving average is equal to the length of the seasonal component. The seasonal component can be estimated by

Definition 7.2.5 Additive

$$\hat{s}_t = x_t - \hat{m}_t$$

Multiplicative

$$\hat{s}_t = \frac{x_t}{\hat{m}_t}$$

[...] many series are dominated by a trend and/or a seasonal effect [...] A simple additive decomposition model is given by

$$x_t = m_t + s_t + z_t$$

where, at time t , x_t is the observed series, m_t is the trend, s_t is the seasonal effect, and z_t is an error term that is, in general, a sequence of correlated random variables with mean zero.

If the seasonal effect tends to increase as the trend increases, a multiplicative model may be more appropriate

$$x_t = m_t s_t + z_t$$

Definition 7.2.6

Once we have identified any trend and seasonal effects, we can deseasonalise the time series and remove the trend. If we use the additive decomposition method, we first calculate the seasonality adjusted time series and then remove the trend by subtraction. This leaves the random component, but the random component is not necessarily well modelled by independent random variables. In many cases, consecutive variables will be correlated. If we identify such correlations, we can improve our forecast, quite dramatically if the correlations are high.

Second Order Properties

A possible approach to forecast future values of a time-series based variable is to extrapolate the current trend and to apply some adaptive estimations.

Exponential Smoothing

Definition 7.2.7 Let's x and y two time series. The cross covariance function of x and y as a function of a lag k , denoted $\gamma(x, y)$, is defined as:

$$\gamma(x, y) = E$$

Definition 7.2.8

Autocorrelation, Cross-correlation and Partial Autocorrelation

Another important feature of most time series is that observations close together in time tend to be correlated (serially dependent)

two unrelated time series will be correlated if they both contain a trend

Autocorrelation measures the (Pearson) correlation of a time series with a delayed version of itself, and as a function of that delay. Autocorrelation is intended to estimate the degree of similarity of an observation with respect to previous observations.

Definition 7.2.9 Let $\{\mathbf{X}_t\}$ be a time series with mean μ and variance σ^2 . The *autocorrelation* function, denoted by ρ , is defined as:

$$\rho_x(k) = \frac{E[(x_t - \mu)(x_{t+k} - \mu)]}{\sigma^2}$$

The value k is called *lag*.

The autocorrelation function is not defined for all time series, because the mean may not exist (time series with a trend), or the variance may be zero (constant time series). A time series for which the autocorrelation is defined is called *second order stationary*.

The *sample autocorrelation* is computed in practice by:

$$\hat{\rho}_x(k) = \frac{\frac{1}{n} \sum_{t=1}^{n-k} (x_t - \bar{x})(x_{t+k} - \bar{x})}{\left(\frac{1}{n} \sum_{t=1}^n (x_t - \bar{x})\right)^2}$$

On the contrary of what happened with autocorrelation, sample autocorrelation is defined in case of time series with a trend. However, we must be carefull about the interpretation of the results. In general, sample autocorrelation is applied over the residuals of a time series once the trend and the seasonal components have been removed.

A correlogram is a plot of the sample autocorrelations $\hat{\rho}(k)$ versus time lags k (see Figure XXX). The dotted lines are drawn at $-\frac{1}{n} \pm \frac{2}{\sqrt{n}}$. If $\hat{\rho}(k)$ is outside these lines for a value of k we have evidence against the null hypothesis that $\hat{\rho}(k) = 0$ at the 5% level (See Section XXX). It is expected that 5% of the estimates $\hat{\rho}(k)$ fall outside these lines.

■ **Example 7.5** TODO: Insert Figure. If the example is drawn using `matplotlib.pyplot.acorr`, the above paragraph is not true. Investigate how the shared areas in the correlogram used by `matplotlib` are computed. ■

Crosscorrelation measures ...

Definition 7.2.10 Let $\{x_t\}$ and $\{y_t\}$ be time series with means μ_x and μ_y and variances σ_x^2 and σ_y^2 . The *crosscorrelation* function, denoted by ρ , is defined as:

$$\rho_{x,y}(k) = \frac{E[(x_{t+k} - \mu_x)(y_t - \mu_y)]}{\sigma_x \sigma_y}$$

The value k is called *lag*.

If the crosscorrelation is defined it is said that the combined model is *second order stationary*. The *sample crosscorrelation* is computed in practice by:

$$\hat{\rho}_{x,y}(k) = \frac{\frac{1}{n} \sum_{t=1}^{n-k} (x_t - \bar{x})(y_{t+k} - \bar{y})}{\frac{1}{n} \sum_{t=1}^n (x_t - \bar{x}) \frac{1}{n} \sum_{t=1}^n (y_t - \bar{y})}$$

In general, sample crosscorrelation is applied over the residuals of both time series once trends and the seasonal components have been removed.

In the same way we draw a correlogram we can plot a crosscorrelogram of the crosscorrelation between two time series.

■ **Example 7.6** TODO: Insert Figure. If the example is drawn using `matplotlib.pyplot.acorr`, the above paragraph is not true. Investigate how the shared areas in the correlogram used by `matplotlib` are computed. ■

Structural Time Series

A univariate structural time series model is one which is formulated in terms of components which, although unobservable, have a direct interpretation. It not only provides the basis for making predictions of future observations, but it also provides a description of the salient features of a time series.

Examples of structural components could be a trend, a seasonal effect, a cycle, an intervention, or the noise.

Definition 7.2.11 Let x_t be a time series. The structural decomposition of x_t is given by

$$y_t = \mu_t + \psi_t + \gamma_t + \dots + \varepsilon_t$$

where $\mu_t, \psi_t, \gamma_t, \dots$ is a finite collection of additive stochastic terms, called structural components, and ε_t is a random term composed by independent and identically distributed samples with zero mean.

■ **Example 7.7** dd ■

State Space Model

Definition 7.2.12 Let y_t be a time series. The state space decomposition of y_t is given by

$$\begin{aligned} y_t &= \mathbf{Z}_t \alpha_t + \varepsilon_t \\ \alpha_{t+1} &= \mathbf{T}_t \alpha_t + \mathbf{R}_t \eta_t \end{aligned}$$

where

- α_t explain
- \mathbf{Z}_t explain
- \mathbf{T}_t explain

$$\mathbf{y} = f(\mathbf{X}) + \varepsilon \quad (7.2)$$

measurement equation and transition equation

Multivariate Time Series

TODO: Introduce this section.

A time series could be multivariate, where a dependant variable $\{x_{m+1}\}$ is sampled together with a collection of m independent variables $\{\mathbf{x}_i\}$.

Definition 7.2.13 A *multivariate time series* of length $n \in \mathbb{N}$ composed by $m+1 \in \mathbb{N}$ variables, denoted by $\{\mathbf{x}_t^i : i = 1, \dots, m+1 \text{ and } t = 1, \dots, n\}$ or $\{\mathbf{x}_t\}$, is a sequence $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n\}$ of *observed vector values*. The time series $\{\mathbf{x}^{m+1}\}$ is called the *independent variable*, and the time series $\{\mathbf{x}^i : i = 1, \dots, m\}$ the *dependant variables*.

TODO: cross-correlation and partial cross-correlation.

7.3 Minimum Message Length

The *Minimum Message Length* (MML) is based on the idea that a good theory, or explanation, for a dataset is a small collection of premises under which the data is not surprising. The best theories are those which are short, able to explain most of the data, and with a high accuracy. An *explanation message* is composed by two parts: the first part comprises all the premises induced from the data, including numerical values; the second part contains all the data that cannot be derived from the premises. The message also assumes the existence of some already known and accepted premises (prior knowledge). Given the prior premises and the message it should be possible to recover the original dataset. According to the MML, theories are not rejected due to contradictory measurements, they only make the second part of the message longer.

In the MML principle, a message is a lossless encoded version of the original data. The first part of the message contains a probabilistic model about the data, and the second part is the data encoded using this model. We are interested in finding the shortest possible explanation message. If the length of the explanation message is longer than the original data, the theory is considered unacceptable.

Bayes' theorem (see Theorem ??) states that the probability $P(H | E)$ of a hypothesis H given an evidence E is:

$$P(H | E) = \frac{P(E | H)P(H)}{P(E)}$$

We are interested in finding the hypothesis H with the highest posterior probability $P(H | E)$ of being true, assuming a fixed evidence E . That is, we are looking for maximize $P(E | H)P(H)$ or, equivalently, maximize $P(H \wedge E)$.

The *Minimum Message Length* principle (MML for short) is based on the idea that the length of encoding $H \wedge E$ as a binary string using an optimal code is equal to $-\log_2 P(H \wedge E)$ (see Theorem 5.3.2). That is:

$$l(H \wedge E) = -\log_2 P(H \wedge E) = -\log_2 P(E | H)P(H) = -\log_2 P(E | H) - \log_2 P(H)$$

The most probable model H would be that model that allows us to encode $H \wedge E$ with the shortest possible string. The encoded string would be composed by two parts, a general assertion about the data, and a detailed description of the data assuming that the assertion is true.

Let \mathbb{X} be a discrete set composed by all possible datasets, \mathcal{X} a random variable taking values on \mathbb{X} , and $f(X | \theta)$ a probability distribution for \mathcal{X} given the parameter θ .

Definition 7.3.1 Let Θ be a discrete set of possible parameters for f with probability distribution $h(\theta)$, $\theta \in \Theta$, and let $\hat{\theta} \in \Theta$ be an inferred parameter. An *assertion* is the encoded version of $\hat{\theta}$ using an optimal code given the probability distribution h .

The length of the assertion given an optimal binary code is $-\log_2 h(\hat{\theta})$ (see Section 5.3). Note that θ could be a single scalar or a vector of values. Moreover, θ could be related to more than one family of probability distributions f .

Definition 7.3.2 Let $X \in \mathbb{X}$ be a dataset, and $\hat{\theta} \in \Theta$ an inferred value for the distribution f . A *detail* is the encoded version of X using an optimal code given the probability distribution $f(X | \hat{\theta})$.

The length of the detail given an optimal binary code is $-\log_2 f(X | \hat{\theta})$. That is, the length of the detail is the negative of the log-likelihood of X given $\hat{\theta}$.

Definition 7.3.3 Let $X \in \mathbb{X}$ be a dataset, and $\hat{\theta} \in \Theta$ an inferred value for the distribution f . A *message* for the dataset X given an inference $\hat{\theta} \in \Theta$ is the concatenation of the assertion for $\hat{\theta}$ and the corresponding detail for X given $\hat{\theta}$.

The length of a message given an optimal binary code is $-\log_2 h(\hat{\theta}) - \log_2 f(X | \hat{\theta})$. The length of a message allow us, for example, to compare the posterior probabilities of two competing explanations or hypotheses $\hat{\theta}_1$ and $\hat{\theta}_2$.

Definition 7.3.4 Let $X \in \mathbb{X}$ be a dataset. The *Minimum Message Length* of X , denoted by $MML(X)$, is given by:

$$MML(X) = \arg \min_{\hat{\theta} \in \Theta} (-\log_2 h(\hat{\theta}) - \log_2 f(X | \hat{\theta}))$$

In practice, the actual messages will not be constructed, since our interest is in the length of the messages, not in their content. It is assumed that the sets \mathbb{X} and Θ and the functions $f(X | \theta)$ and $h(\theta)$ are known a priori, and so, it is not necessary to include them as part of our encoding message.

■ **Example 7.8** Consider an experiment in which we toss a weighted coin 100 times. Denote by 1 if we get a face and 0 a cross, so that each experiment is a binary string of length 100. Our collection of all possible datasets is $\mathbb{X} = \mathcal{B}^{100}$, θ is a number in the interval $[0, 1]$, the likelihood $f(X | \theta)$ follows a binomial distribution (that is, $f(X | \theta) = \theta^n(1 - \theta)^{100-n}$ where n is the number of faces in X), and since we do not know anything about how the coin is weighted, we could assume that $h(\theta)$ is the uniform distribution in the interval $[0, 1]$ (that is, $h(\theta) = 1$ for $\theta \in \Theta$). Under these assumptions, the length of a message for X given an inferred parameter $\hat{\theta}$ would be:

$$-\log_2 h(\hat{\theta}) - \log_2 f(X | \hat{\theta}) = -n \log_2 \hat{\theta} - (100 - n) \log_2 (1 - \hat{\theta})$$

We are interested in finding the value of $\hat{\theta}$ that minimizes the length of the encoded version of X , that is, the minimum message length for X . ■

A Maximum A Posteriori analysis of the experiment in Example 7.8 would provide the same inference for $\hat{\theta}$ than the Minimum Message Length approach. Moreover, given that $h(\theta)$ follows an uniform distribution, a Maximum Likelihood approach would reach exactly the same value for $\hat{\theta}$.

7.4 Minimum Description Length

The *Minimum Description Length* (MDL) principle is a reformulation of the Kolmogorov complexity with the goal to make it applicable to solve practical problems. MDL explicitly address the two most important practical limitations of the Kolmogorov complexity: its uncomputability (in general, the Kolmogorov complexity cannot be computed), and the large constants involved in the invariance theorem (that makes it inapplicable to short strings). The approach of MDL to these problems is to scale down Kolmogorov complexity until it does become applicable: instead of using general-purpose computer languages, MDL is based on fixed languages and coding functions.

In contrast to Kolmogorov that states that the complexity of a string is equal to the length of the shortest program that prints that string, MDL proposes that our capacity to learn about a string is equivalent to our ability to compress that string. The idea behind MDL is to describe a dataset with the help of an hypothesis: the more the hypothesis fits the data (and here good fit equals learning), the more we can compress the data given that hypothesis.

In our particular case, we are interested in MDL because it will allow us to compute the nescience of a topic given a dataset, instead of requiring a text describing the topic. Thus, given a topic t and a sample dataset $D = \{x_1, x_2, \dots, x_n\}$, the nescience of a particular hypothesis H will be related to the capacity of that hypothesis to compress the dataset.

MDL comes into two versions, *simplified (two-part code) MDL* and *refined MDL*. Simplified MDL is easier to understand, and it will allows us to introduce some important concepts and notation. The extension of the concept of nescience to datasets will be based on the refined version of MDL.

TODO: Explain how this relates to cross entropy

In this section we are going to introduce the two-part code version of the minimum description length principle for probabilistic models.

Given a set of candidate models (a set of probability distributions (for example first-order Markov chains) or functions of the same functional form (for example the kth degree polynomials)) $\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \dots$ the simplified, two-part version, of the Minimum Description Length Principle [Gr=0000FC05] states that the best point hypothesis (a single probability distribution (e.g. a Markov chain will all parameters values specified) $H \in \mathcal{H}^{(1)} \cup \mathcal{H}^{(2)} \cup \dots$ to explain the data $D = (x_1, \dots, x_n) \in \mathcal{X}^n$ is the one that minimizes the sum $L(M) + L(D | M)$, where $L(M)$ is the length (in bits) of the model description, and $L(D | M)$ is the length (in bits) of the data encoded with the help of the hypothesis ... there is only one reasonable choice for this code ... the so-called Shannon-Fano code ... Each hypothesis H may be viewed as a probability distribution over \mathcal{X}^n .

For each such distribution there exists a code with length function L such that for all $x^n \in \mathcal{X}^n$ we have that $L(x^n) = -\log_2 P(x^n | H)$. The quantity $L(M)$ depends on each model.

A description of the data “with the help of” a hypothesis means that the better the hypothesis fits the data, the shorter the description will be. A hypothesis that fits the data well gives us a lot of information about the data. Such information can always be used to compress the data. This is because we only have to code the errors the hypothesis makes on the data rather than the full data.

The sum of the two description length will be minimized at a hypothesis that is quite (but not too) “simple”, with a good (but not perfect) fit.

The length of the data given the model, that is, $L(D | M)$.

$$-\log P(D) = L_C(D)$$

This choice for C gives a short codelegth to sequences which have high probability according to (k, ttita) while it gives a high codelength to sequences with low probability. The codelength thus directly reflects the goodness-of-fit of the data with respect to (k, tita) measured in terms of the probability fo D according to (k, tita) .

When we say we “code the data D with the help of probabilistic hypothesis P” we mean that we code D using the Shannon-Fano code corresponding to P.

$$L(D | P) := -\log P(D)$$

the code with these lengths is the only one that would be optimal if P where true. (mention we are only interested in code lengths, we are not interested in to find the code itself).

For $L(M)$ we use the standard code for integers.

■ **Example 7.9** Markov Chain Hypothesis Selection: Suppose we are given data DX^n where $X = \{0, 1\}$. We seek a reasonable model for D that allows us to make good predictions of future data coming from the same source. We decide to model our data using the cass B o all Markov chains [...] we face the problem of overfitting: for a given sequence $D = (x_1, \dots, x_n)$, there may be a Markov chain P of very high order that fits data D quite well but that will behave very badly when predicting future data from the same source. ■

7.4.1 Refined MDL

In refined MDL, we associate a code for encoding D not with a single $H \in \mathcal{H}$ but with the full model \mathcal{H} ... we design a single one-part code with lengths $\bar{L}(D | H)$ (called the stochastic complexity of the data given the model). This code is designed such that whenever there is a member of (parameter in) \mathcal{H} that fits the data well, in the sense the $L(D | H)$ is small, then the codelenth $\bar{L}(D | H)$ will also be small. Codes with this property are called universal codes.

There are at least four types of universal codes:

- 1 The normalized maximum likelihood (NML) code and its variations.
- 2 The Bayesian mixture code and its variations.
- 3 The prequential plug-in code.
- 4 The two-part code.

Refined MDL is a general theory of inductive inference based on universal codes that are designed to be minimax, or close to minimax optimal. It has mostly been developed for model selection, estimation and prediction.

7.5 Multiobjective Optimization

Multiobjective optimization is the area of mathematics that deals with the problem of simultaneously optimizing two or more conflicting functions. Multiobjective optimization has been applied in many areas of science, including engineering, economics and logistics, where there is no single solution that simultaneously satisfies all objectives, so a decision must be made in the presence of trade-offs between the conflicting goals.

From a formal point of view, we are interested in solving the following *multiobjective optimization* problem:

$$\begin{aligned} \text{minimize } & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})\} \\ \text{subject to } & \mathbf{x} \in \mathbf{S} \end{aligned}$$

where $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$, $i = 1, \dots, k$, are two or more objective *objective functions*, and the nonempty set $\mathbf{S} \subset \mathbb{R}^n$ is the *feasible region*, whose elements $\mathbf{x} = (x_1, x_2, \dots, x_n)$ are *decision vectors*. The image of the feasible region $f(\mathbf{S}) \subset \mathbb{R}^k$, denoted by \mathbf{Z} , is called *objective region*, and its elements $\mathbf{z} = (f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x}))$ *objective vectors*. In some applications, the feasible region is formed by a collection of inequality constraints $\mathbf{S} = \{\mathbf{x} \in \mathbb{R}^n \mid g(\mathbf{x}) = (g_1(\mathbf{x}), \dots, g_m(\mathbf{x})) \leq 0\}$.

In this book we will be dealing with nonlinear multiobjective minimization problems, where at least one of the objective functions, or the constraint functions, is not linear. Objective functions can be also incommensurable, that is, measured in different units or in different scales.

Since the objective functions are conflicting, it does not exist a single solution that is optimal with respect to every objective function (the objective region is partially ordered).

Definition 7.5.1 A decision vector $\mathbf{x} \in \mathbf{S}$ *dominates* another decision vector $\mathbf{y} \in \mathbf{S}$ if $f_i(\mathbf{y}) \leq f_i(\mathbf{x})$ for all $i \in \{1, \dots, k\}$ and $f_j(\mathbf{y}) < f_j(\mathbf{x})$ for at least one $j \in \{1, \dots, k\}$. An objective vector $\mathbf{z} \in \mathbf{Z}$ *dominates* another objective vector $\mathbf{w} \in \mathbf{Z}$ if $w_i \leq z_i$ for all $i \in \{1, \dots, k\}$ and $w_j < z_j$ for at least one $j \in \{1, \dots, k\}$.

Dominance can be studied from the point of view of decision variable space or objective space. An objective vector dominates another objective vector if, and only if, its corresponding decision vector also dominates the other decision vector.

We are interested in those objective vectors for which none of its individual components can be improved without deteriorating at least one of the others.

Definition 7.5.2 A decision vector $\mathbf{x} \in \mathbf{S}$ is *Pareto optimal* if there does not exist another decision vector $\mathbf{y} \in \mathbf{S}$ such that \mathbf{y} dominates \mathbf{x} . An objective vector $\mathbf{z} \in \mathbf{Z}$ is Pareto optimal if there does not exist another objective vector $\mathbf{w} \in \mathbf{Z}$ such that \mathbf{w} dominates \mathbf{z} .

Pareto optimality can be also studied from the point of view of decision variable space or objective space. An objective vector is Pareto optimal if, and only if, its corresponding decision vector is Pareto optimal.

Definition 7.5.3 The set of Pareto optimal solutions, denoted by \mathbf{P}_D , is called the *Pareto optiomal set*. The set of Pareto optimal solutions in the space of objectives, denoted by \mathbf{P}_O , is called the *Pareto frontier*.

Sometimes, in practice, it is convenient to use a more restrictive definition of the concept of optimaility, in which we identify those vectors for which there does not exist any other vector that improves over all the components simultaneously.

Definition 7.5.4 A decision vector $\mathbf{x} \in \mathbf{S}$ is *weakly Pareto optimal* if there does not exist another decision vector $\mathbf{y} \in \mathbf{S}$ such that $f_i(\mathbf{y}) < f_i(\mathbf{x})$ for all $i = 1, \dots, k$. An objective vector $\mathbf{z} \in \mathbf{Z}$ is weakly Pareto optimal if there does not exist another objective vector $\mathbf{w} \in \mathbf{Z}$ such that $w_i < z_i$

for all $i = 1, \dots, k$.

An objective vector is weakly Pareto optimal if its corresponding decision vector is weakly Pareto optimal. Obviously, the Pareto optimal set is a subset of the weakly Pareto optimal set.

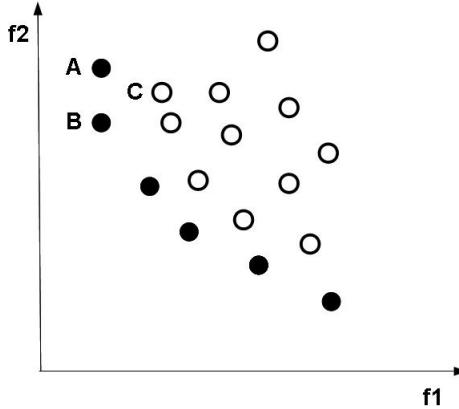


Figure 7.1: Pareto optimality.

■ Example 7.10 TODO: Provide an example based on a multi-variable function. In figure 7.1 we have depicted a sample of the objective region of a multiobjective optimization problem composed by two real-valued objective functions f_1 and f_2 that we are interested in minimizing. White points are not weakly Pareto optimal since there exist points that improve both components at the same time (for example, point **B** improves point **C** in both functions). Black points are weakly Pareto optimal since there is no point that improves both components at the same time. Point **A** is not Pareto optimal since point **B** improves one component without deteriorating the other. ■

Mathematically speaking all the solutions that compose the Pareto optimal set are equally good. However, for the majority of the practical applications, it is highly desirable to have a single solution. Finding this solution requires additional information not included in the definition of the optimization problem. The relation of preference between objective function values is expressed using a *decision maker*, that it is supposed to have additional insights about the problem to solve. In this sense, solving a multiobjective optimization problem would require to find those feasible decision vectors that are Pareto optimal and that satisfy the additional requirements imposed by the decision maker.

In practice, we assume that the preferences of the decision maker can be expressed using a value function.

Definition 7.5.5 A *value function* is a function $U : \mathbb{R}^k \rightarrow \mathbb{R}$ that assigns to each objective vector $\mathbf{z} = (z_1, \dots, z_k)$ a single real value $U(\mathbf{z})$.

value functions are maximized

Value functions allow us to order the vectors of the objective region \mathbf{Z} . We are interested in applying the value function to the Pareto optimal subset to find a unique solution to the multiobjective optimization problem.

7.5.1 Range of the Solutions

We are interested in investigating the range of the solutions included in the Pareto optimal set. To do that, we have to find the lower and upper bounds of this set. In the rest of this section, we assume that the objective functions are bounded over the feasible region \mathbf{S} .

TODO: rewrite the concepts of nadir and ideal vector using the supremum and infimum.

An objective vector that minimizes all objective functions is called an ideal objective vector.

Definition 7.5.6 A vector $\mathbf{z}^* \in \mathbb{R}^k$ is called *ideal* if each of its components z_i , minimizes the objective function $f_i(\mathbf{x})$ subject that $\mathbf{x} \in \mathbf{S}$.

If there exists an ideal objective vector that belongs to the feasible region, that is $\mathbf{z}^* \in \mathbf{Z}$, then that vector would be a solution of the optimization problem, and that solution would be unique. Ideal vectors are lower bounds to the Pareto set.

The upper bound of the Pareto optimal set is given by the nadir objective vector. The nadir vector can be estimated using the decision vectors calculated when obtaining the objective ideal vector.

Definition 7.5.7 Let $\mathbf{z}^* \in \mathbb{R}^k$ be an ideal objective vector. The set of decision vectors $\{\mathbf{x}_1^*, \dots, \mathbf{x}_k^*\}$ used to compute \mathbf{z}^* is called the *payoff table* for \mathbf{z}^* . That is, if $\mathbf{z}^* = \{z_1^*, \dots, z_k^*\}$, we have that $z_i^* = f_i(\mathbf{x}_i^*)$.

It turns out that $f_i(\mathbf{x}_i^*)$ ($i = 1, \dots, k$) is minimal for all for all the elements of the payoff table.

Having the payoff table we can provide an estimation for the nadir vector. For simplicity, let's f_{ij}^* denote the value of the objective function f_i computed over the vector \mathbf{x}_j^* , where $i, j = 1, \dots, k$.

Definition 7.5.8 Let $\mathbf{z}^* \in \mathbb{R}^k$ be an ideal objective vector, and $\{\mathbf{x}_1^*, \dots, \mathbf{x}_k^*\}$ its payoff table. The *nadir* objective vector $\mathbf{z}^{nad} = \{z_1^{nad}, \dots, z_k^{nad}\}$ is given by $z_i^{nad} = \max_j f_{ij}^*$.

The ideal objective vector and the nadir objective vector may, or may not, be feasible. In figure 7.2 are depicted the ideal (vector **E**) and nadir (vector **F**) vectors of the multiobjective optimization problem of Example 7.10.

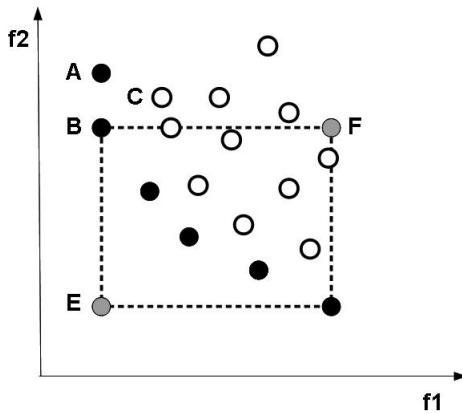


Figure 7.2: Ideal and Nadir vectors.

For some applications, the range of values of the objective functions can differ by orders of magnitude. In those situations, it is advisable to normalize them, so the values are in the same scale. We can use the ideal and nadir vector for this normalization process, by replacing each objective function $f_i(\mathbf{x})$ ($i = 1, \dots, k$) by the normalized function

$$\frac{f_i(\mathbf{x}) - z_i^*}{z_i^{nad} - z_i^*}$$

7.5.2 Trade-offs

Since the functions we want to minimize are conflicting, sometimes we have to assume that the only way to gain a benefit in one aspect of the problem is to lose something in another aspect. How much we have to give up in one objective to improve a certain quantity in the other is called a trade-off.

Definition 7.5.9 Let $\mathbf{x}^1, \mathbf{x}^2 \in \mathbf{S}$ be two decision vectors. The *ratio of change* between the functions f_i and f_j for the vectors $\mathbf{x}^1, \mathbf{x}^2$, denoted by Δ_{ij} , is defined as:

$$\Delta_{ij}(\mathbf{x}^1, \mathbf{x}^2) = \frac{f_i(\mathbf{x}^1) - f_i(\mathbf{x}^2)}{f_j(\mathbf{x}^1) - f_j(\mathbf{x}^2)}$$

for all $i, j = 1, \dots, k$ such that $f_j(\mathbf{x}^1) - f_j(\mathbf{x}^2) \neq 0$.

Δ_{ij} is called a *partial trade-off*, involving f_i and f_j between \mathbf{x}^1 and \mathbf{x}^2 if $f_l(\mathbf{x}^1) = f_l(\mathbf{x}^2)$ for all $l = 1, \dots, k, l \neq i, j$. If $f_l(\mathbf{x}^1) \neq f_l(\mathbf{x}^2)$ for at least one $l = 1, \dots, k$, and $l \neq i, j$ then Δ_{ij} is called a *total trade-off*.

If the trade-off between two objective functions is very small or very large, that is, if a small change in one aspect of the optimization problem has a significant impact in another aspect, we have a case that is similar to having a weakly Pareto solution that it is not Pareto optimal. In some practical applications is convenient to filter out those solutions that present this undesirable behaviour.

Definition 7.5.10 A decision vector $\mathbf{x} \in \mathbf{S}$ is *properly Pareto optimal* if it is Pareto optimal and if there exists a real number $M > 0$ such that for each f_i and $\mathbf{y} \in \mathbf{S}$ satisfying $f_i(\mathbf{y}) < f_i(\mathbf{x})$, there exists at least one f_j such that $f_j(\mathbf{x}) < f_j(\mathbf{y})$ and

$$\frac{f_i(\mathbf{x}) - f_i(\mathbf{y})}{f_j(\mathbf{y}) - f_j(\mathbf{x})} \leq M$$

An objective vector $\mathbf{z} \in \mathbf{Z}$ is properly Pareto optimal if the decision vector corresponding to it is properly Pareto optimal.

A solution is properly Pareto optimal if there is at least one pair of objectives functions for which a small decrement in one objective is possible only at the expense of a large increment in the other objective.

Note that the properly Pareto optimal set is a subset of the Pareto optimal set, and the Pareto optimal set is a subset of the weakly Pareto optimal set.

7.5.3 Optimization Methods

Generating Pareto optimal solutions plays an important role in multiobjective optimization, and mathematically the problem is considered to be solved when the Pareto optimal set is found [...] However, this is not always enough. We want to obtain only one solution. This means that we must find a way to put the Pareto optimal solutions in a complete order. This is why we need a decision maker and his preference structure.

In general, multiobjective optimization problems are solved by scalarization [...] scalarization means converting the problem into a single or a family of single objective optimization problems with a real-valued objective function

- three requirements are set for a scalarization function: 1) It can cover any Pareto optimal solution.
- 2) Every solution is Pareto optimal.

[...] classify the methods according to the participation of the decision maker in the solution process. The classes are: 1) methods where no articulation of preference information is used (no-preference methods) 2) methods where a posteriori articulation of preference is used (a posteriori methods) 3) methods where a priori articulation of preference information is used (a priori methods), and 4) methods where progressive articulation of preference information is used (interactive methods).

In no-preference methods, the knowledge of the decision maker is not taken into account, and the optimization problem is solved using a relatively simple method. In a posteriori methods,

the set (or part of it) of Pareto optimal points is identified and then the decision maker select the preferred solution among the alternatives.

Global Criterion

The global criterion is a non-preference method in which the distance between some reference point and the feasible objective region is minimized. In this method, all the objective functions are considered to be equally important. As reference point it is usually used the ideal vector, and as metric it is common to use a L_p -metric. Under these assumptions, the global criterion method becomes the following minimization problem:

$$\begin{aligned} \text{minimize} \quad & \left(\sum_{i=1}^k (f_i(\mathbf{x}) - z_i^*)^p \right)^{\frac{1}{p}} \\ \text{subject to} \quad & \mathbf{x} \in \mathbf{S} \end{aligned}$$

Different values for p result in different solutions to the minimization problem. Common values for p are 1, 2 or ∞ .

Proposition 7.5.1 The solution of the L_p -based global criterion is Pareto optimal.

Proof. Let \mathbf{x} be a solution of the L_p -based global criterion problem, with $1 \leq p < \infty$, and assume that \mathbf{x} is not Pareto optimal. Then, according to Definition 7.5.2 there must exist a point $\mathbf{y} \in \mathbf{S}$ such that $f_i(\mathbf{y}) \leq f_i(\mathbf{x})$ for all $i = 1, \dots, k$, and $f_j(\mathbf{y}) < f_j(\mathbf{x})$ for at least one j . Then we have that $(f_i(\mathbf{y}) - z_i^*)^p \leq (f_i(\mathbf{x}) - z_i^*)^p$ for all $i \neq j$ and $(f_j(\mathbf{y}) - z_j^*)^p < (f_j(\mathbf{x}) - z_j^*)^p$. Adding all these terms and raising to the $1/p$ power, we obtain

$$\left(\sum_{i=1}^k (f_i(\mathbf{y}) - z_i^*)^p \right)^{\frac{1}{p}} < \left(\sum_{i=1}^k (f_i(\mathbf{x}) - z_i^*)^p \right)^{\frac{1}{p}}$$

which is a contradiction with the fact that \mathbf{x} is a solution to the minimization problem. ■

Although all the solutions selected by the L_p -based global criterion are Pareto optimal, as we have proved in previous proposition, there are solutions of the optimal Pareto set that will never be selected by this method. In practice it is convenient to normalize the range of the objective values, so that those points closer to the ideal vector do not receive more importance. A common normalization term used in practice is $z_i^{nad} - z_i^*$.

Weighting Method

The weighting method is a simple way to generate different Pareto optimal solutions.

In the weighting method [...] the idea is to associate each objective function with a weighting coefficient and minimize the weighted sum of the objectives. In this way, the multiple objective functions are transformed into a single objective function. We suppose that the weighting coefficients w_i are real numbers such that $w_i \geq 0$ for all $i = 1, \dots, k$. It is also usually supposed that the weights are normalized, that is $\sum_{i=1}^k w_i = 1$.

The multiobjective optimization problem is modified into the following problem, to be called a weighting problem:

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^k w_i f_i(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathbf{S} \end{aligned}$$

where $w_i \geq 0$ for all $i = 1, \dots, k$ and $\sum_{i=1}^k w_i = 1$.

[...] weighting coefficient zero makes no sense. It means that we have included in the problem some objective function that has no significance at all.

The objective functions should be normalized or scaled so that their objective values are approximately of the same magnitude [...] Only in this way can one control and manoeuvre the method to produce solutions of a desirable nature in proportion to the ranges of the objective functions. Otherwise the role of the weighting coefficients may be greatly misleading.

Proposition 7.5.2 The solution of weighting problem is Pareto optimal if the weighting coefficients are positive, that is $w_i > 0$ for all $i = 1, \dots, k$.

Proof. TODO ■

the solution of the weighting method is always Pareto optimal if the weighting coefficients are all positive or if the solution is unique [...] The weakness of the weighting method is that not all of the Pareto optimal solutions can be found.

the weighting method is used to generate Pareto optimal solutions

in practical calculations the condition $w_i \geq \epsilon$, where $\epsilon > 0$, must be used instead of the condition $w_i > 0$ for all $i = 1, \dots, k$. This necessitates a correct choice as to the value of ϵ .

The weighting method can be used so that the decision maker specifies a weighting vector representing his preference information [...] In this case, the weighting problem can be considered (a negative of) a value function (remember that value functions are maximized).

If the weighting method is used as an a priori method one can ask what the weighting coefficients in fact represent. Often, they are said to reflect the relative importance of the objective functions. However, it is not at all clear what underlies this notion [...] instead of relative importance, the weighting coefficients should represent the rate at which the decision maker is willing to trade off values of the objective functions

if some of the objective functions correlate with each other, then seemingly "good" weighting vectors may produce poor results and seemingly "bad" weighting vectors may produce useful results

Weighting coefficients are not easy to interpret and understand for average decision makers.

Employing the weighting method as an a priori method presumes that the decision maker's underlying value function is or can be approximated by a linear function [...] it must be noted that altering the weighting vectors linearly does not have to mean that the values of the objective functions also change linearly. It is difficult to control the direction of the solutions by the weighting coefficients

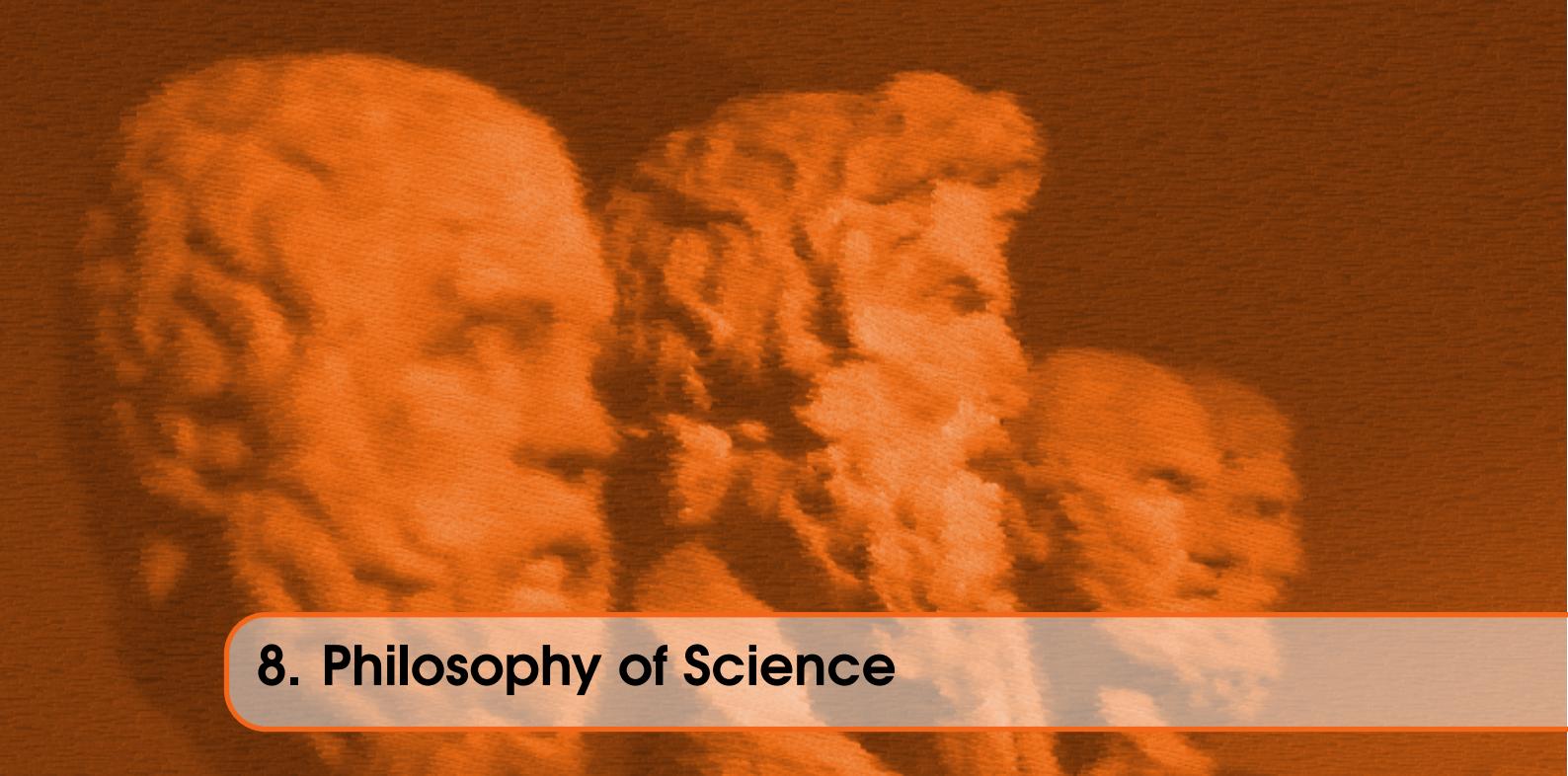
Shall I explain the ϵ -Constraint method?

References

TODO: Add paper of Turing about AI.

The minimum message length principle was developed by Chris Wallace, published for first time in 1968 in [wallace1968information]. The book [wallace2005statistical], written by the same author, contains a detailed description of the principle.

A good introduction to the discipline of non-linear multiobjective optimization can be found in [miettinen2012nonlinear]; Section 7.5 is largely based on this book.



8. Philosophy of Science

*To go where you don't know,
you have to go the way you don't know.*
San Juan de la Cruz

Warning: This section still requires a significant amount of work!

In this Chapter we provide a quick review of all those elements from Philosophy that are relevant to the theory of nescience. In particular, from the area of Philosophy of Science.

Although philosophy does not provides tools we can use in practice in our theory, it provides the tools to analysis the products of the theory. Moreover, philosophy is very helpful in order to ask the right questions.

8.1 Metaphysics

Metaphysics is the branch of philosophy that studies the nature of things in very general terms. Metaphysics deals with concepts like substance, properties, change, time, causes, ... There are two basic kinds of entities: particulars and properties. We know about things in the world through their properties. Properties might change, but the particular to which they attach remain.

Clarify in terms of particulars, properties and relations.

According to the substratum point of view, particulars have to be something other than its properties, and so, we have to abstract away the properties. A substratum would be that thing that underlies the properties and holds them all together in one place. However, it is extremely difficult to describe the nature of this substratum that hold all these properties.

The bundle theory proposes that an object is only of a collection (or bundle) of properties appropriately arranged, there is no substratum supporting those properties. This theory has the advantage that we do not need to explain what it is the substratum, but it present also problems, for example, there is no way to distinguish between two particulars that have exactly the same properties, since they should be the same thing. Particulars cannot be, or cannot be reduced, to their properties. How do we deal with change. If a property changes, a property is lost, or a new one is

gained, we have a different collection of properties, is it still the same thing? We could argue that properties change with time, but things remain numerically the same.

Abstract things, abstract particulars, have a very different kind of nature.

The Platonic realm is a transcendent world, above and beyond the physical world. This world of ideas contains all the true version of all the properties and relations. It can only be contemplated and understood through pure intellect. Things in the world are imperfect copies of this heavenly world. It is a strong form of realism, since things are more real than the imperfect copies of our world.

Anti-realism denies this world of ideas, because there is no way to bring together the world of Forms and the real world. Everything is down here in the Earth. Mathematics are just definitions that do not exist.

Nominalism propose that properties are just names, words used to describe groups of particular things that resemble each other. There are no properties, only particulars.

8.2 Scientific Representation

Science allows us to learn about the world, and this learning process is made through the use of representations of research entities. Examples of such representations include measurements with scientific instruments, descriptions based on texts, digital pictures like X-rays or MRI scans, etc. Also, in science, we usually consider as valid representations mathematical equations, models and theories. The problem of *scientific representation* deals with the necessary and sufficient conditions for being a scientific representation, and to identify the desired properties of good representations. Problems addressed in this philosophical discipline include:

- *Scientific Representation Problem*: Which are the sufficient and necessary conditions that a representation has to satisfy in order to be considered a valid scientific representation? Do these conditions depend on the particular science or the context of research?
- *Representational Demarcation Problem*: Are scientific representations different from other types of representations? How do they differ?
- *Problem of Style*: Given that the same entity can be represented in many different ways, what types, or styles, of representations are there? Which are their characteristics? Are they fixed, or new styles can be invented?
- *Standard of Accuracy*: What constitutes an accurate representation? How do we distinguish between accurate and non-accurate representations?
- *Problem of Ontology*: What kind of objects can be used as representations? Are representations concrete or abstract? Do we require that representations have to be realistic?

There are also five conditions of adequacy that a scientific representation should satisfy:

- *Requirement of Directionality*: Representations describe entities in the real world, but how do real world entities describe their representations?
- *Surrogate Reasoning*: How representations allow us to generate hypothesis about their targets?
- *Applicability of Mathematics*: How mathematical models represent the real world?
- *Possibility of Misrepresentation*: Are non-accurate representations also valid representations?
- *Targetless Models*: Do we allow representations that do not represent anything?

There have been multiple proposals to formally define the concept of scientific representation. Unfortunately, none of these proposals can provide a convincing answer to the questions and conditions of adequacy described above. In the rest of this section we describe some of these proposals, identifying their advantages and drawbacks. In order to compare these proposals we will declare them as "A scientific model M represents a target system T if, and only if ...".

The *Stipulative Fiat* states that "a scientific model M represents a target system T if, and only if, a scientist stipulates that M represents T ." The main problem with this interpretation is that,

since anything can be a representation if a scientist say so, how we can guarantee the surrogate reasoning condition? The proponent of this theory admits that some representations are more useful than other.

The *Similarity conception* proposes that "a scientific model M represents T if, and only if, M and T are similar." This conception solves the surrogate reasoning condition, since being similar we can derive similar properties. However, it introduces new challenges, being the problem of style the main one. There are also issues with directionality and accuracy. In which sense are they similar?

The *Structuralist conception* is based on the concept of isomorphism, that is, a scientific model M represents a target system T if the structure of M is isomorphic to the structure of T . Having the same structure justify the surrogate reasoning, and given that mathematics is the study of structures, justify the applicability of math in nature. By structure we mean a relation over a set.

Inferential conception: A model M is an epistemic representation of a certain target T if and only if the user adopts an interpretation of M in terms of T .

The Fiction View of Models: M is a scientific representation of T iff M functions as prop in game of make-believe which prescribes imagining about T .

Representation-As:

8.3 Models in Science

Models are one of the main tools we have today to do science. From an ontological point of view, there is a large variety of things that can be considered as models. Models can be physical objects, for example, scaled down (or scaled up) pieces made of wood or metal, a wood model of a car; they can be also fictional, that is, abstract ideas residing in the mind of scientists, like Bohr model of the atom. Mathematical models, either set theoretic structures [...] example [...] or equations, like the Black-Scholes partial differential equations to estimate the price of some financial derivative products like options. Finally, we could also consider models descriptions, like for example, the ones included in scientific papers.

From a semantic point of view, models can be also representations of target systems, in the sense already covered in the previous section. In this sense, models have the same problems already covered. And they have to be considered for each type of model (physical, fictional, mathematical, ...). An idealized model is a deliberate simplification of something complicated with the objective of making it more tractable [...] Aristotelian idealization amounts to 'stripping away' [...] all properties from a concrete object that we believe are not relevant to the problem at hand [...] Galilean idealizations are ones that involve deliberate distortions [...] point masses [...]. In mathematical models we have also approximations.

[...] epistemology [...] how do we learn with models? [...] Models are vehicles for learning about the world [...] models allow for surrogate reasoning [...]

Learning about a model happens at two places, in the construction and the manipulation of the model [...] There are no fixed rules or recipes for model building and so the very activity of figuring out what fits together and how affords an opportunity to learn about the model. Once the model is built, we do not learn about its properties by looking at it; we have to use and manipulate the model in order to elicit its secrets [...] by performing though experiment [...] An importatn class of models is of mathematical nature [...] solve equations analytically [...] making a computer simulation.

Once we have knowledge about the model, this knowledge has to be 'translated' into knowledge about the target system [...] there do not seem to be any general accounts of how the transfer of knowledge from a model to its target is achieved

[...] philosophy of science [...] how do models relate to theory? Models and other debates

8.4 Scientific Theories

8.5 The Scientific Method

The study of the scientific method is the attempt to discern the activities by which [science is an enormously success]. Among the activities often identified as characteristic of science are systematic observation and experimentation, inductive and deductive reasoning, and the formation and testing of hypotheses and theories [...] methods are the means by which [the goals of science] are achieved [...] methodological rules are proposed to govern method [...] method is distinct [...] from the detailed and contextual practices through which methods are implemented [...] how pluralist do we need to be about method? [...] how much can method be abstracted from practice? [...] Unificationists continue to hold out for one method essential to science; nihilism is a form of radical pluralism, which considers the effectiveness of any methodological prescription to be so context sensitive as to render it not explanatory on its own.

[...] scientific activity varies so much across disciplines, times, places, and scientists that any account which manages to unify it all will either consist of overwhelming descriptive detail, or trivial generalization [...] For most of the history of scientific methodology the assumption has been that the most important output of science is knowledge and so the aim of methodology should be to discover those methods by which scientific knowledge is generated [...] very few philosophers arguing any longer for a grand unified methodology of science

On the hypothetico-deductive account, scientist work to come up with hypotheses from which true observational consequences can be deduced.

A distinction in methodology was made between the contexts of discovery and of justification. The distinction could be used as a wedge between, on the one hand the particularities of where and how theories or hypotheses are arrived at and, on the other, the underlying reasoning scientists use [...] when assessing theories and judging their adequacy on the basis of the available evidence. By and large [...] philosophy of science focused on the second context.

[...] the Hypothetico-Deductive (H-D) method [...] a theory [...] is confirmed by its true consequences

Method may therefore be relative to discipline, time or place [...] by the close of the 20th century the search by philosophers for the scientific method was flagging.

A problem with the distinction between the contexts of discovery and justification [...] is that no such distinction can be clearly seen in scientific activity [...] new scientific concepts are constructed as solutions to specific problems by systematic reasoning, and that of analogy, visual representation and thought-experimentation are among the important reasoning practices employed [...] model-based reasoning consists of cycles of construction, simulation, evaluation and adaptation of models that serve as interim interpretations of the target problem to be solved [...] this process will lead to modifications or extensions, and a new cycle of simulation and evaluation [...] there is no logic of discovery [...] a large and integral part of scientific practice is [...] the creation of concepts through which to comprehend, structure, and communicate about physical phenomena [...] science as problem solving [...] scientific problem solving as a special case of problem-solving in general [...] the primary role of experiments is to test theoretical hypotheses according to the H-D model [...] exploratory experimentation was introduced to describe experiments driven by the desire to obtain empirical regularities and to develop concepts and classifications in which these regularities can be described [...] the development of high throughput instrumentation [...] has given rise to a special type of exploratory experimentation that collects and analyses very large amounts of data [...] data-driven research.

[...] the ability of computers to process, in a reasonable amount of time, huge quantities of data [...] computers allow for more elaborate experimentation [...] but also, through modelling and simulations, might constitute a form of experimentation themselves [...] does the practice of using computers fundamentally change scientific method, or merely provide a more efficient means

fo implementing standar methods? [...] Because computers [...] many of the steps involved in reaching a conclusion on the basis of experiment are nore made inside a "black box" [...] we ought to consider computer simulan a "qualitatively different way of doing science" [...] simulation as a "third way" for scientific methodology (theoretical reasoning and experimental practice are the first two ways)

[...] a fixed four or five step procedure starting from observations and description of a phenomenon and progressing over formulation of a hypothesis which explains the phenomenon, designing and conducting experiments to test the hypothesis, analyzing the results, and ending with drawing a conclusion [...] conclusion of recent philosophy of science that there is not any unique, easily described scientific method.

8.6 Scientific Discovery

Scientific discovery refers to the process of conceiving new scientific ideas, hypotheses or novel explanations. Scientific discovery involves an "eureka moment" or "happy though" in which the new idea is sough, its formal articulation, and the validation process. In this section, we are interested in the fist part of this process, that is, the eureka moment. We are interested in the nature of this insightful moment, and in particular, if it can be analyzed, and if there exists rules, algorithms, guidelines, or heuristics, to generate these novel insights. We do not consider in this section if those hypotheses are worth articulating and testing.

explain that during this epoch, doing sience and meta-science was the same activity During the 17th and 18th centuries, great philosophers, like Bacon, Descartes and Newton proposed methods to discover new knowledge. Briefly describe their ideas

[...] the two pocesses of conception and validation of an idea or hypothesis became distinct, and the view that the merit of a new idea does not depend on the way in which it was arrived at became widely accepted.

The general agreement among philosophers is that the creative process of conceiving a new idea is a non-rational process that can not be formalized as a set of steps. Current philosophy of science focus on the formulation and justification of new ideas rather than finding them. because philosophy of science is intended to be normative

Discovery as abduction [...] the act of discovery [...] follows a distinctive logical pattern, which is different from both inductive logic and the logic of hypothetico-deductive reasoning. The special logic of discovery is the logic of abductive or "retroductive" inferences [...] an inference beginning with [...] surprising or anomalous phenomena [...] discovery is primarily a process of explaining anomalies or surprising, astonishing phenomena. The scientists' reasoning proceeds abductively from an anomaly to an explanatory hypothesis in light of which the phenomena would no longer be surprising or anomalous [...] the schema of abductive reasoning does not explain the very act of conceiving a hypothesis or hypothesis-type.

Heuristic programming [...] artificial intelligence at the intersection of philosophy of science and cignitive science [...] problem solving activity [...] whereby the systematic aspects of problem solving are studied within an information-processing framework. The aim is to clarify with the help of computational tools the nature of the methdos used to discover scientific hypothesis [...] searches for solutions [...] "problem space" in a certain domain [...] the basic idea behind computational heuristics is that rules can be identified that serve as guidelines for finding a solution to a given problem quickly and efficiently by avoiding undesired states of the problem space [...] the data from actual experiments the simulations cover only certain aspects of scientific discoveries [...] they do not design newe expeirments, instrumets, or methods [...] the complex problem spaces for scientific problems are often ill defined

In recent decades, philosophers have subsumed their interest in this eureka moment. However, the research is no only philosophy based, they borrow ideas and collaboate, with areas like

cognitive science, neuroscience, computational research, and environmental and social psychology, philosophers have sought to demystify the cognitive processes involved in the generation of new ideas

[...] A discovery is not a simple act, but an extended, complex process, which culminates in paradigm changes. Paradigms are the symbolic generalizations, metaphysical commitments, values, and exemplars that are shared by a community of scientists and that guide the research of that community [...] A discovery begins with an anomaly, that is, with the recognition that the expectations induced by an established paradigm are being violated.

Some authors have tried to define exactly what we mean by being creative, proposing that it is novel, surprising and important.

[...] the role of analogy in the development of new knowledge, whereby analogy is understood as a process of bringing ideas that are well understood in one domain to bear on a new domain [...] the distinction between positive, negative and neutral analogies [...] the distinction between horizontal and vertical analogies between domains.

Model-based reasoning proposes that much of the human problem solving is based on mental models rather than the application of the laws of logic to a collection of propositions. According to this theory, human mind uses model based representations to visualize how the world works, and to manipulate the structure of these models, using tools like analogy, or thought experiments. Unfortunately, the concept of model is too vague.

The formal methods of scientific discovery are covered in Section XXX.

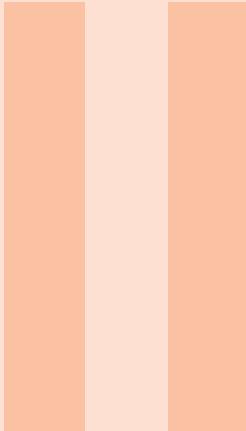
Psychological studies of creative individuals' behavioral dispositions suggest that creative scientists share certain personality traits, including confidence, openness, dominance, independence, introversion, as well as arrogance and hostility [...] creative individuals usually have outsider status - they are socially deviant and diverge from the mainstream

References

Add entry of Scientific Representation in the Stanford Encyclopedia of Philosophy

Perhaps add entries for Lewis Carroll's Sylvie and Bruno and Borges' On Exactitude in Science

?? contains an interesting review of the concept of science, the scientific method, and the role that technology plays in our society. The author proposes that the goal of science should be quality, although the concept of quality is left undefined, and how to reconcile the rational and romantic points of view in science. The book also contains some advice about which is the right state of mind to pursue a scientific problem, and how to deal with the inevitable failures.



9.1	Entities	
9.2	Representations	
9.3	Invalid Representations	
9.4	Joint Representations	
9.5	Descriptions	
9.6	Descriptions for Joint Representations	
9.7	Conditional Descriptions	
9.8	Research Areas	
9.9	References	
Part 2. Foundations		
10	Miscoding	169
10.1	Miscoding	
10.2	Joint Miscoding	
10.3	Decreasing Miscoding	
10.4	Targetless Representations	
10.5	Miscoding of Areas	
11	Inaccuracy	177
11.1	Inaccuracy	
11.2	Conditional Inaccuracy	
11.3	Decreasing Inaccuracy	
11.4	Inaccuracy-Miscoding Rate of Change	
11.5	Inaccuracy of Areas	
12	Surfeit	187
12.1	Surfeit	
12.2	Conditional Surfeit	
12.3	Decreasing Surfeit	
12.4	Rate of Change	
12.5	Surfeit of Areas	
13	Mismodel	193
13.1	Mismodel	
13.2	Reducing Mismodel	
13.3	Mismodel of Areas	
14	Nescience	197
14.1	Nescience	
14.2	Minimizing Nescience	
14.3	Joint Nescience	
14.4	Conditional Nescience	
14.5	Nescience of Areas	
14.6	Perfect Knowledge	
14.7	Current Best Description	
14.8	Nescience based on Datasets	
14.9	Unknown Unknown	
15	Interesting Questions	209
15.1	Relevance	
15.2	Applicability	
15.3	Interesting Questions	
15.4	New Research Topics	
15.5	Classification of Research Areas	
16	Advanced Properties	219
16.1	The Axioms of Science	
16.2	Scientific Method	
16.3	The Inaccuracy - Surfeit Trade-off	
16.4	Science vs. Pseudoscience	
16.5	Graspness	
16.6	Effort	
16.7	Human Understanding	
16.8	Areas in Decay	



9. Entities, Representations and Descriptions

*We are all agreed that your theory is crazy.
The question which divides us is whether it is crazy enough.*
Niels Bohr

The initial step towards quantifying our lack of knowledge involves a precise identification of the research entities under examination. The specific components of this collection depend largely on the specific use case of the nescience theory. Each application necessitates a unique assortment of entities, be it mathematical objects, living organisms, human needs, and so forth. Fortunately, the process of quantifying how much we do not know remains consistent across all cases.

The subsequent step involves devising a methodology to represent the identified entities through strings of symbols, or what we term as 'representations'. Properly symbolizing a research entity is a challenging, yet unresolved, epistemological problem. The solution suggested within the context of nescience theory is rooted in the concept of an oracle Turing machine. The ease of implementing this solution in practice depends largely on the abstract nature of the entities under study. For instance, encoding the entirety of abstract mathematical objects presents significant difficulty, necessitating the discovery of an approximation. In contrast, encoding all possible computer programs (considering our focus on software quality, refer to Chapter 18) is relatively straightforward, given that computer programs inherently exist as strings.

The final step, once a suitable method of encoding the original entities as string-based representations has been established, requires producing a description. This should be as accurate and concise as possible, detailing our current understanding of these representations. In nescience theory, it's imperative that these descriptions are computable - that is, a computer should be capable of fully recreating the original representation based on the description. A challenge arises with descriptions as they characterize representations, i.e., the encoding of entities, not the entities themselves. Therefore, the effectiveness of a description in representing an entity is contingent upon the quality of the representation used.

In this chapter, we will formalize all these key concepts: entities, representations, descriptions, among others. We will also explore what constitutes a perfect description, how to compute the

combined representation of multiple entities, and how to describe a representation assuming another's perfect description. We will further examine the concept of a research area and its properties.

9.1 Entities

Defining the nature of a research entity is an unresolved, intricate philosophical problem. We tackle this complexity from a fundamentally pragmatic perspective. Our theory initiates from the premise that there exists a non-empty collection of *entities* that we aim to comprehend.

Notation 9.1. *We represent the set of research entities of interest as \mathcal{E} .*

The exact constituents of \mathcal{E} are contingent on the specific domain in which the theory of nescience is employed, typically aligning with a specific area of knowledge. Instances of entity sets might include: research components in mathematics (abstract); the kingdom of Animalia (living entities); both known and unknown human needs (abstract); all potential computer programs (strings), and so forth. In a strict sense, \mathcal{E} is not a well-defined set, given that it is generally indeterminate what qualifies as a member of this set.

The abstract character of \mathcal{E} affords certain benefits but also imposes significant constraints. The primary constraint is that our definition of nescience is, in most instances, a non-computable quantity, necessitating practical approximations. The chief benefit is that the novel concepts and methods introduced in this book can be applied across a spectrum of problems, not solely restricted to the pursuit of new scientific knowledge.

In the theory of nescience, we preclude the allowance of universal sets, that is, we cannot suppose the existence of a set ξ encompassing everything. The issue with universal sets is that they contravene Cantor's theorem (refer to Example 9.1). Cantor's theorem establishes that the power set $\mathcal{P}(\xi)$, constituted by all possible subsets of ξ , possesses more elements than the original set ξ . This contravenes the assumption that ξ incorporates everything. In nescience theory, the set \mathcal{E} must be indicative of a specific set.

■ **Example 9.1 — Cantor's theorem.** Cantor's theorem proposes that for any set A , $d(A) < d(\mathcal{P}(A))$. Consider $f : A \rightarrow \mathcal{P}(A)$, a function that maps each element $x \in A$ to the set $\{x\} \in \mathcal{P}(A)$; evidently, f is injective, implying $d(A) \leq d(\mathcal{P}(A))$. To substantiate that the inequality is strict, let's assume there exists a surjective function $g : A \rightarrow \mathcal{P}(A)$ and consider the set $B = \{x \in A : x \notin g(x)\}$. As g is surjective, there must exist a $y \in A$ such that $g(y) = B$. This, however, raises a contradiction, $y \in B \Leftrightarrow y \notin g(y) = B$. Consequently, the function g cannot exist, therefore $d(A) < d(\mathcal{P}(A))$. ■

In the theory of nescience, not all conceivable sets are acceptable, as some can lead to paradoxes. Take for example, Russell's paradox, which proposes a set R composed of all sets that are not members of themselves. The paradox emerges when we attempt to discern whether R is a member of itself (refer to Example 9.2). To circumvent such predicaments, the theory of nescience is founded on the Zermelo-Fraenkel set of axioms, accompanied by the Axiom of Choice (see Appendix C). Russell's paradox is attributed to an abuse of the set builder notation $\{\cdot\}$. The *axiom of separation* (if P is a property with parameter p , then for any set x and parameter p there exists a set $y = \{u \in x : P(u)\}$ that includes all those sets $u \in x$ having property P) permits the use of this notation only to construct sets that are subsets of already existing sets. A broader *axiom of comprehension* (if P is a property, then there exists a set $y = \{u : P(u)\}$) would be required to permit sets like the one proposed by Russell's paradox. In the ZFC axioms, and within the theory of nescience, the axiom of comprehension is deemed to be false.

■ **Example 9.2 — Russell's Paradox.** Suppose R is the set of all sets not members of themselves, such that $R = \{x : x \notin x\}$. The contradiction arises when querying if R is a member of itself. If

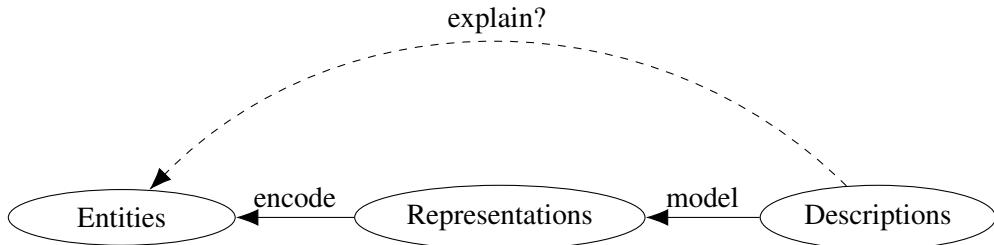


Figure 9.1: The Problem of Understanding

R is not a member of itself, by its own definition, it should be; conversely, if R is declared to be a member of itself, its definition dictates it should not be. Symbolically, this can be written as $R \in R \Leftrightarrow R \notin R$. ■

In nescience theory, we do not delve into classical ontological queries, i.e., the classification of existent entities in the world that can be known. Additionally, we do not strive to resolve epistemological issues, such as how scientific knowledge is authenticated through evidence or the nature of said evidence.

Once a set \mathcal{E} of entities is chosen, the subsequent step involves uniquely encoding these entities as strings of symbols, otherwise, their description would be impossible (unless the entities themselves are symbol strings). The method to effectively execute this encoding is delineated in the subsequent section.

9.2 Representations

The conceptualization and depiction of entities within a set \mathcal{E} , which may be abstract in nature, pose significant epistemological challenges that have engaged researchers for over two millennia (refer to Section 8.2 for a succinct overview of ongoing issues and unresolved queries). This book outlines a potential resolution to this complex problem, while analyzing its merits and shortcomings. Our proposed approach bifurcates the issue of scientific representation into two interconnected subproblems: the modeling of descriptions into representations, and the encoding of entities into these representations. Thus, scientific descriptions serve to define entities through an indirect association facilitated by representations (refer back to Figure 1.1 in the book's introduction, which has been reproduced in this section for convenient referencing). The current section will primarily concentrate on the aspect of encoding, with Section 9.5 focusing on descriptions.

Within the field of Kolmogorov complexity, the representation issue is tackled by presupposing that the set \mathcal{E} is well-defined, countable, and that a total encoding function $f : \mathcal{E} \rightarrow \mathbb{N}$ exists, mapping the set of entities to the set of natural numbers (akin to Gödel numbering). The nescience theory similarly embraces this concept of encoding entities via numbers, or, in other words, strings of symbols. However, our approach to the problem of encoding an arbitrary set \mathcal{E} involves an inversion of the problem. This involves defining a total function $\mathcal{O}_{\mathcal{E}} : \mathcal{S}^* \rightarrow \mathcal{E}$ that maps the well-defined set \mathcal{S}^* (comprising all possible finite strings from an alphabet \mathcal{S}) to the potentially non-countable and non-well defined set \mathcal{E} containing the entities of interest.

Our function $\mathcal{O}_{\mathcal{E}}$, which is contingent upon the set \mathcal{E} , operates much like an oracle (taking inspiration from the concept of an oracle Turing machine as per Definition 4.5.1). This function has the capacity to entirely reconstruct an entity $\mathcal{O}_{\mathcal{E}}(x) \in \mathcal{E}$ using its encoding string $x \in \mathcal{S}^*$, without the need for additional information. Evidently, this oracle is an abstract concept and its real-world construction would be implausible for most of the sets \mathcal{E} . However, the theoretical existence of this oracle assists in the elucidation and verification of key attributes of the scientific discovery process. Due to its oracle-like nature, $\mathcal{O}_{\mathcal{E}}$ exhibits limitations regarding the mathematical operations it can

be employed in, as will be elucidated in the subsequent discourse.

Our premise rests on the assumption of the existence of a singular, distinct physical world that remains uninfluenced by observers. Furthermore, we posit that for certain collections of entities \mathcal{E} , it is logically coherent to presuppose the existence of an oracle $\mathcal{O}_{\mathcal{E}}$ (refer to the discussion on the scientific representation problem in Section 8.2). For the purposes of this book, without any loss of generality, we will exclusively regard binary strings as the means of encoding entities.

Definition 9.2.1 Given \mathcal{E} as a set of entities, a *representation function* is defined as an oracle $\mathcal{O}_{\mathcal{E}} : \mathcal{B}^* \rightarrow \mathcal{E}$ that maps elements of the set \mathcal{B}^* , which comprises all conceivable finite binary strings, to the elements of the set \mathcal{E} , corresponding to the entities under examination.

Only the elements of the set \mathcal{B}^* , that is, binary strings, can serve as representations of entities (problem of ontology). We do not allow drawings, or any other form of physical models, unless they are converted into binary strings. We do not make any distinction between scientific representations and any other kind of representations (demarcation problem). It is up to the oracle to decide which entity is encoded by each representation. It is also important to note that the oracle $\mathcal{O}_{\mathcal{E}}$ is total, that is, all possible strings represent entities (no targetless models allowed).

Within this framework, only the elements of the set \mathcal{B}^* , that is, binary strings, are eligible to serve as representations of entities (problem of ontology). We do not permit other forms of physical models, including drawings, unless they can be transcribed into binary strings. We do not distinguish between scientific representations and other forms of representations (demarcation problem). The oracle assumes the responsibility of determining which entity is encoded by each representation. It is also pivotal to note that the oracle $\mathcal{O}_{\mathcal{E}}$ is total, meaning all potential strings represent entities (models without targets are not permissible).

■ **Example 9.3** For a given set of entities \mathcal{E} , the existence of a representation oracle $\mathcal{O}_{\mathcal{E}}$ does not imply its uniqueness. For instance, a binary negation oracle (which transforms the zeros of a binary string into ones, and the ones into zeros), assigning to each string $x \in \mathcal{B}^*$ the entity $\mathcal{O}_{\mathcal{E}}(\neg x)$, would also qualify as a representation function. ■

Rather than deploying individual oracles, we could have employed a universal oracle machine. This is a machine $\mathcal{U}_{\mathcal{O}}$ which, given the encoding of an oracle $\mathcal{O}_{\mathcal{E}}$ and a string s as input, computes $\mathcal{U}_{\mathcal{O}}(\langle \mathcal{O}_{\mathcal{E}}, s \rangle) = \mathcal{O}_{\mathcal{E}}(s)$. Universal machines are particularly applicable to the universal set ξ , encompassing all entities. However, such an approach would likely complicate the process of scientific discovery in practical terms. We elect instead to work with sets of entities \mathcal{E} corresponding to distinct areas of knowledge, choosing a single oracle $\mathcal{O}_{\mathcal{E}}$ for each set \mathcal{E} (the most suitable one according to our current understanding).

■ **Example 9.4** Consider the instance when the subjects of study are animals. Initially, we could utilize detailed physical descriptions of the animals as encodings. In this scenario, the oracle would be a hypothetical machine capable of recreating the original animal from its description. As our comprehension of the biology of life advances, we could contemplate the adoption of an alternative encoding based on the animals' DNA. Both these encodings serve as valid representations of the entities. ■

As illustrated in Example 9.3 and Example 9.4, the entities of \mathcal{E} could be encoded in various manners (problem of style). Different oracles will accommodate different encoding schemas. Some strings offer superior representations of an entity compared to others (standard of accuracy). The optimal representation is contingent on the type of queries we seek to answer. Employing a specific style of encoding with an oracle that anticipates a different style may yield unforeseen results. Our objective should be to utilize representations that minimize the size of the oracle (i.e., the volume of inherent knowledge presumed to be contained in the oracle).

■ **Example 9.5** Consider \mathbf{X}_t as a time series comprising m measurements x_1, \dots, x_m gathered at fixed intervals from a physical phenomenon exhibiting sinusoidal behavior, and let's assume we have trained a multi-layer perceptron neural network nn that perfectly fits the data, that is, it returns x_t when given a time t as input. An analysis of the time series' cycles will lead us to discern that the most appropriate model for this particular physical phenomenon appears to be a sine function. Nonetheless, no extant machine learning methodology would reach the same conclusion when provided with the architecture of the neural network (number of layers, sizes, and trained weights) as input. Clearly, the oracle is so sophisticated that it can, indeed, achieve this. ■

Remember that the oracle functions as a total function, implying that every conceivable string must represent an entity, even the incomplete ones. The more information a representation lacks, the more challenging it becomes for us to decipher the functioning of the oracle. It's crucial to exercise caution with representations that don't capture the entirety of the original entities, as the results of our analyses could exhibit bias. Chapter 10 provides an in-depth exploration of errors attributed to the miscoding of abstract entities, illustrating that not only must representations pertain to entities, but also that entities must be relevant to representations (requirement of directionality).

We employ an oracle function as opposed to an oracle relation $\mathcal{R}_{\mathcal{O}} \subset \mathcal{B}^* \times \mathcal{E}$, not with the aim of pairing strings with their corresponding entities, but to construct an entity given its representation. The oracle's capacity to reconstruct original entities legitimizes our ability to formulate hypotheses about entities based on their representations (surrogate reasoning). According to the theory of nescience, scientific investigation involves not only discerning how to encode entities appropriately, but also uncovering the intricacies of the decoding oracles.

The purpose of encoding entities in the theory of nescience differs from that of Shannon's information theory, as illustrated in Example 9.6.

■ **Example 9.6** Take, for instance, a set \mathcal{E} comprising two books: "The Ingenious Nobleman Sir Quixote of La Mancha" and "The Tragedy of Romeo and Juliet". We could encode the first book with the string "0" and the second with the string "1". While these strings allow us to uniquely identify each book in the set, they don't qualify as suitable encodings in the context of the theory of nescience. Information theory is about identifying an object uniquely given a reference, necessitating mutual agreement between the sender and receiver about the mapping between references and objects. On the other hand, in the theory of nescience, we're interested in providing a representation that encapsulates all the nuances and details of the original objects. For instance, it's impossible to hypothesize about Cervantes' influence on Shakespeare's work given the strings "0" and "1". ■

An alternative approach to addressing the problem outlined in Example 9.6, where we used artificially brief descriptions, could involve demanding the set \mathcal{E} to be infinite, as Kolmogorov complexity does (thereby precluding infinite instances of deception). However, even with an infinite set of entities, not every possible encoding schema can ensure the satisfaction of the highly desirable property of surrogate reasoning.

We distinguish between *knowable* and *unknowable* entities.

Definition 9.2.2 We say that an entity $e \in \mathcal{E}$ is *knowable* if there exists at least one $r \in \mathcal{B}^*$ such that $\mathcal{O}_{\mathcal{E}}(r) = e$. An entity $e \in \mathcal{E}$ is *unknowable* if it is not knowable.

A priori, it is not possible to determine if a set of entities \mathcal{E} is knowable, unknowable, or partially knowable. Selecting the right knowable entities to study is a matter of trial and error. In this book we have nothing to say about unknowable entities, beyond that the unknowability of an entity cannot be proved nor discovered.

Definition 9.2.3 Let $e \in \mathcal{E}$ a knowable entity. We call the *set of representations* for e , denoted by \mathcal{R}_e , to $\{r \in \mathcal{B}^* : \mathcal{O}_{\mathcal{E}}(e) = r\}$.

In figure 9.2 we have a representation of an hypothetical oracle from the set of finite binary strings \mathcal{B}^* to the set of entities \mathcal{E} ; we have also depicted a particular entity e_1 , the subset of strings \mathcal{R}_{e_1} that encode that entity, and a particular representation r_1 . Since the set \mathcal{E} is, in general, not well defined, the inverse function $\mathcal{O}_\mathcal{E}^{-1}(e_1)$ cannot be computed in practice.

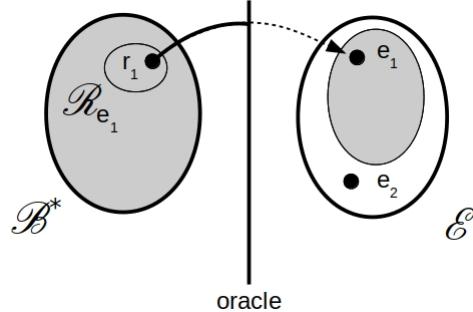


Figure 9.2: Encodings and Entities.

A consequence of working with finite strings as representations is that it might happen that there exist entities that are not encoded by any representation (see the gray areas in Figure 9.2, in particular, the entity e_2 is not encoded by any representation). Intuitively, we could say that for some domains of knowledge the number of problems is much higher than the number of solutions.

■ **Example 9.7** If the collection of entities under study are real numbers, it turns out that there exist numbers that can not be encoded using finite binary strings, since the set \mathbb{R} has the cardinality of the continuum, and the set \mathcal{B}^* is numerable. ■

Since our knowledge about the entities and the inner workings of the oracle are incomplete, in practice we will work with another set of strings that we believe are close to the representations \mathcal{R}_e that encode the entity e . The elements that belong to this set usually change over time, as we better understand the entities of \mathcal{E} , and how the oracle encodes these entities as strings. The more abstract is our set of entities, the more difficult will be to approximate them as strings (see Chapter 10).

■ **Example 9.8** The entity "luminiferous ether" was a theoretical postulate about a hypothetical medium in which the light would propagate. The ether was used as an explanation of how a wave-based light could propagate through the empty space. In 1887, the results of the Michelson-Morley experiment suggested that the ether did not exist, and after Einstein formulated his special theory of relativity, that successfully explained how light propagates through empty space, the idea of ether was completely dropped. ■

If the selected oracle is not minimal, finding what it is exactly a valid representation will require to understand how the oracle works internally. By non-minimal oracle we mean that part of the information required to reconstruct the original entity is hard-wired in the oracle itself, instead of encoded as symbols in the representations. Working with non-minimal oracles could make the research process easier.

R One of the problems of science, and in general of any human intellectual activity, is that people tend to confuse symbols with what they represent. The theory of nescience has been carefully designed to avoid this problem, by means of clearly stating the difference between research entities and the representation of entities. However, keeping this distinction always explicit in the explanations would make the book very difficult to read. We have tried to find a compromise between clarity in the exposition and rigor in the definitions. Sometimes, during the introduction of new ideas we talk in general about a *topics*, meaning an entity, a representation, or both, an entity and its representation. But, in the mathematical definitions

and propositions we always make this difference unequivocal. In case of doubt about what we mean, please take the mathematical definitions as the authoritative reference.

9.3 Invalid Representations

It might happen that the representation we are using for an entity $e \in \mathcal{E}$ is an incorrect one, that is, instead of working with a string $r \in \mathcal{B}^*$ that perfectly encodes e , we are studying another string $r' \in \mathcal{B}^*$ that it is, hopefully, close to r but not necessarily equal. We are interested in to compute the distance between the string r' and the perfect one r as a quantitative measure of the error we are introducing due to the use of a wrong encoding. Unfortunately, we do not know r , since for the majority of the practical applications, it does not exists a computable function from \mathcal{E} to \mathcal{B}^* . The only thing we have to our disposal is an abstract oracle machine $\mathcal{O}_{\mathcal{E}}$ that knows which strings represents each entity. Recall from Section 9.2 that we propose to address the scientific representation problem by introducing an abstract oracle $\mathcal{O}_{\mathcal{E}} : \mathcal{B}^* \rightarrow \mathcal{E}$ from the set \mathcal{B}^* of all possible finite binary strings to the set \mathcal{E} of entities under study.

We start by distinguishing between valid representations, that is strings that contain all the relevant information required by the selected oracle to reconstruct an entity, and non-valid representations.

Definition 9.3.1 Let \mathcal{E} be a collection of entities, and $\mathcal{O}_{\mathcal{E}}$ an encoding function. We define the set of *valid* representations for \mathcal{E} given the encoding function $\mathcal{O}_{\mathcal{E}}$, denoted by $\mathcal{R}_{\mathcal{O}_{\mathcal{E}}}^*$, as the subset of representations that perfectly encode an entity of \mathcal{E} , according to $\mathcal{O}_{\mathcal{E}}$.

The set $\mathcal{R}_{\mathcal{O}_{\mathcal{E}}}^*$ is, in general, unknown, since it is a subset of \mathcal{B}^* whose definition depends on an abstract, not well defined, oracle. Intuitively, a representation is valid if it contains all the details required so that the oracle can reconstruct the original entity without requiring the use of external information. A valid representation cannot contain wrong symbols nor relevant information can be missing, since in that case the oracle would not be able to reconstruct the entity. Moreover, we require that the oracle is not only able to reconstruct an entity given a representation but it can also derive the set of valid representations given an entity. In this sense, an entity that contains non-relevant information cannot be valid, since that part of the string cannot be generated by the oracle given the entity.

■ **Example 9.9** The data used in time of Ptolomeo about the position of celestial bodies along the year was a non-valid encoding of the entity "position of celestial bodies", since the data was inaccurate. A better encoding was the data collected by Tycho Brahe. Today, we have even better encodings. Adding to the dataset the name of the person who collected the data is an example of a representation that includes non-relevant information. ■

Recall that we do not allow other representations of entities except finite binary strings. Drawings, mathematical formulas, or datasets can be used as representations as long as they can be encoded as strings and are recognized as valid representations by the selected oracle. Each string represents one, and only one entity. Of course, different oracles will define different sets of valid representations. In general, we will assume that a particular representation oracle $\mathcal{O}_{\mathcal{E}}$ has been selected to encode the set \mathcal{E} of entities in which we are interested.

Notation 9.2. We will denote the set of valid representations of \mathcal{E} by $\mathcal{R}_{\mathcal{E}}^*$ if there is no ambiguity about which oracle $\mathcal{O}_{\mathcal{E}}$ we are using.

We can define the set of valid representations of a single entity e in the same way.

Definition 9.3.2 We define the set of valid representations for an entity $e \in \mathcal{E}$, denoted by \mathcal{R}_e^* , as the subset of representations that perfectly encode the entity e , that is, $\mathcal{R}_e^* = \mathcal{R}_{\mathcal{E}}^* \cap \mathcal{R}_e$.

As we have seen in Example 9.4 there could be more than one valid representation for some entities, even if the oracle is restricted to a particular style. It might also happen that the set of valid representations is empty, that is, there is no valid representation for that entity. In that case the entity will be unknowable.

Notation 9.3. We denote by $r_e^* \in \mathcal{R}_{\mathcal{E}}^*$ the fact that r is a valid representation of the entity e .

All possible strings represent an entity, even the non-valid representations, like wrong or incomplete ones. A consequence of working with approximations instead of the true representations is that some of the candidate strings currently in use might encode a different entity from what we were expecting.

Example 9.10 In 1961, the Soviet physicist Nikolai Fedyakin, performed a series of experiments resulting in what was seemingly a new form of water. The new water, called polywater, showed a higher boiling point, a lower freezing point, and much higher viscosity than ordinary water. Later experiment showed that polywater was nothing more than contaminated water with small amounts of impurities. ■

We can safely assume (it is free from logical contradictions) that the oracle machine not only knows which representations encode which entities, but also how far is a string r from being a valid representation r_e^* , for all $r \in \mathcal{B}^*$ and all $r_e^* \in \mathcal{R}_{\mathcal{E}}^*$.

If a representaton r is non-valid, the oracle will find the closest possible representation that it is valid, and assume that they represent the same entity.

The oracle defines an equivalence relation that splits the set \mathcal{L} into equivalence classes. Each equivalence class corresponds to an entity of \mathcal{E} , and it might happens that not all entities of \mathcal{E} have an associated class (what we have called the unknowable unknown). All the strings are related to some entity, and some of them would be better than others (i.e. lower nescience). No string can be part of two different entities. For each collection of \mathcal{E} there could be more than one valid oracle.

Proposition 9.3.1 Axioms A1, A2 and A3 state that the oracle \mathcal{O} is an equivalence relation

$$\text{A1 } \forall t \in \mathcal{L} t \mathcal{O} t.$$

$$\text{A2 } \forall s, t \in \mathcal{L} \text{ if } s \mathcal{O} t \text{ then } t \mathcal{O} s.$$

$$\text{A3 } \forall r, s, t \in \mathcal{L} \text{ if } r \mathcal{O} s \text{ and } s \mathcal{O} t \text{ then } r \mathcal{O} t.$$

Definition 9.3.3 Let \mathcal{L}/\mathcal{O} be the quotation set defined by the oracle relation over the language set. We call *entity*, denoted by $[e]$, to every class of this quotation set, that is, $[e] \in \mathcal{L}/\mathcal{O}$.

9.4 Joint Representations

We have seen in the previous section that there exists more than one way to encode an entity $e \in \mathcal{E}$, what we have called the set \mathcal{R}_e of representations of the entity. Some of these representations have a high quality, in the sense that they contain all the information required by the oracle to reconstruct (in whatever way the oracle manages to do that) the original entity. But also, in the set \mathcal{R}_e there exists low quality representations, that is, representations that lack many of the details needed to fully reconstruct the entity. Furthermore, the set \mathcal{R}_e can contain representations that include information that is wrong, symbols that the oracle will automatically ignore during the reconstruction, but that they will mislead us when understanding the entity. In Chapter 10 we will study how to measure the error due to incomplete and wrong representations, and in this section we will introduce a mechanism to improve our currently known representations.

If we want to increase our knowledge about an entity, we have to find the best possible representation for that entity, that is, a representation that is complete and correct. One way to do that is simply try different strings until we come up with a high quality representation. However, that method could require a lot of time and it is impractical. A more efficient approach would be to complement our current bad representation with more (potentially missing) symbols, or by combining those known representations that contain partial information. These two methods require the introduction of the concept of joint representation.

Definition 9.4.1 Let $s, t \in \mathcal{B}^*$ be two different representations. We call *joint representation* of s and t to the concatenation string st .

■ **Example 9.11** TODO: use an example in which we talk about concatenating two representations, but do not mention the quality of the individual and joint representations (this is covered in the chapter of miscoding) Suppose that the research entity e in which we are interested is the causes of lung cancer. In order to understand this entity, we have measured a collection of risk factors in a random sample of the population (smoking, exercise, diet, age, etc.). However, due to a problem with the sampling procedure, all the samples correspond to a subset of the population, for example, males. This dataset would be a representation s for our entity e , but a very bad one, since it is strongly biased. If we have a second representation, corresponding the sample data of females t , the joint representation st will be a better one than any of them, s or t , isolated. ■

The concatenation of any two arbitrary representations is also a representation.

Proposition 9.4.1 Let $s, t \in \mathcal{B}^*$ be two different representations and st its joint representation, then st is also a representation.

Proof. As we have seen in Section 2.2 the set \mathcal{B}^* is closed under the operation of concatenation of strings, and the representation function $\mathcal{O}_{\mathcal{E}}$ is total. ■

We do not require the set of representations \mathcal{R}_e for a given entity $e \in \mathcal{E}$ to be closed under the operation of concatenation. That is, it might happen that $st \notin \mathcal{R}_e$, even if it is the case that $s \in \mathcal{R}_e$ and $t \in \mathcal{R}_e$.

The concept of joint representation is defined for every pair of strings $s, t \in \mathcal{B}^*$, even if they do not belong to the same set of representations \mathcal{R}_e , in other words, when $\mathcal{O}_{\mathcal{E}}(s) \neq \mathcal{O}_{\mathcal{E}}(t)$. This process of joining representations from different entities will be very useful for the discovery of new research entities hitherto unknown (see Section 15.4 and recall that all possible strings in \mathcal{B}^* represent an entity).

The operation of string concatenation is associative, that is, $(rs)t = r(st)$, for all $r, s, t \in \mathcal{B}^*$. This property defines the algebraic structure of the sets of representations.

Proposition 9.4.2 The set \mathcal{B}^* of representations together with the operation of concatenation has the structure of free monoid.

Proof. As we have seen in Section 2.2 the operation of string concatenation in \mathcal{B}^* is associative, and the empty string λ plays the role of neutral element. ■

We do not require the operation of joining representations to be commutative with respect to the oracle function. Given the representations $s, t \in \mathcal{B}^*$ it might happen that the concatenation strings st and ts represent different entities, that is, $\mathcal{O}_{\mathcal{E}}(st) \neq \mathcal{O}_{\mathcal{E}}(ts)$.

The concept of joint representation can be extended to any arbitrary, but finite, collection of representations. In this way, we could add multiple partial representations to our research, or use them in the process of discovering new entities.

Definition 9.4.2 Let $r_1, r_2, \dots, r_n \in \mathcal{B}^*$ be a finite collection of representations. We call *joint representation* of r_1, r_2, \dots, r_n to the string $r_1r_2\dots r_n$.

It is easy to show that $r_1r_2\dots r_n \in \mathcal{B}^*$ for all $r_1, r_2, \dots, r_n \in \mathcal{B}^*$, that is, \mathcal{B}^* is closed under the operation of concatenation of multiple, finite, representations.

We have seen that the contatenation of two or more representations of an entity can encode a different entity than the original one. This can even happen when the two representations to join encode the same entity.

multiple representations of the same entity can help us to find a better representation. We can take advantage of this property to discover new, previously unknown entities, by means of combining the representations of the already known entities. Intuitively, we are looking for new entities by creating new representations that are different from the representations we already know.

TODO: Formalize this idea with a proposition.

Provide an example of how it can be applied in practice.

9.5 Descriptions

So far, our aim with the strings of \mathcal{B}^* has been to provide an enconding, or representation, as complete and detailed as possible of the entities of \mathcal{E} , no matter its length. However, as we have said in the preface of this book, human understanding requires the derivation of concise models for those entities, since human reasoning cannot be based on long representations.

■ **Example 9.12** In Example 10.4 we have shown that a good representation for the entity "lung cancer" could be a sample dataset in which we measure different risk factors. If smokers decide to quit smooking is not because they know and understand this large sample dataset, but because they know and understand the much simpler derived model "smooking increases the risk of lung cancer". ■

A description or model¹ is a finite binary string mapped to a representation of an entity (recall Figure 1.3 from Chapter 1). Descriptions do not model entities (target systems) directly, they do so through string based representations. In the theory of nescience we require that descriptions must be computable, so we can fully and effectively reconstruct the original representations given their descriptions. The requirement of computability allows us to clearly state the limits of the concept of "description". For example, the problem of self-referential descriptions, like the Berry paradox, can be addressed in the scope of the limits of computation.

Definition 9.5.1 — Model. Let $d \in \mathcal{B}^*$ be a binary string in the form $d = \langle TM, a \rangle$, where TM is the encoding of a prefix free Turing machine and a is the input string to that machine. If $TM(a)$ is defined, we say that d is a *description*.

Intuitively, a description is composed by two parts, a Turing machine that should compresses all the regularities found in this representation, and a string containing what is left, that is, the non-compressible part.

Definition 9.5.2 We define the *set of descriptions*, denoted by \mathcal{D} , as:

$$\mathcal{D} = \{d \in \mathcal{B}^* : d = \langle TM, a \rangle \wedge TM(a) \downarrow\}.$$

Let $r \in \mathcal{B}^*$ be a representation. We define the set of *descriptions for r*, denoted by \mathcal{D}_r , as:

$$\mathcal{D}_r = \{d \in \mathcal{D} : TM(a) = r\}.$$

¹In the theory of nescience use the words "description" and "model" interchangeably.

Finally, given an entity $e \in \mathcal{E}$, we define the set of *descriptions for e*, denoted by \mathcal{D}_e , as:

$$\mathcal{D}_e = \{d \in \mathcal{D} : \exists r \in \mathcal{R}_e, TM(a) = r\}.$$

From an ontological point of view, descriptions are just string based representations that satisfy the additional requirement of being computable. In this sense, descriptions are also representations, and so, there exists descriptions that describe descriptions. In practice it is not a good idea to use descriptions as the representation of entities, since what we are looking for in a good representation is that they contain as many details as possible about the original entities, not a concise encoding. Working with descriptions in the role of representations would make the job of scientific discovery very difficult for humans.

Since each description describes one, and only one, representation, we can define a function that maps descriptions into representations. Given that descriptions are Turing machines, it is natural to use as description function a universal Turing machine. Consequently, not only the individual descriptions of representations are computable, but also the function that maps descriptions into representations is also computable.

Definition 9.5.3 We call *description function*, denoted by δ , to any universal Turing machine $\delta : \mathcal{D} \rightarrow \mathcal{B}^*$ that maps descriptions to their corresponding representations.

If $d = \langle TM, a \rangle$ is a description of the representation r , then we have that $\delta(d) = \delta(\langle TM, a \rangle) = TM(a) = r$.

Inspired by the Occam's razor principle², if two explanations are indifferent, we should prefer the shortest one. Therefore, the limit of what can be known (understand) about a representation, that is, its perfect model, is given by the shortest description that allows us to reconstruct this representation.

Definition 9.5.4 Let \mathcal{D}_r be the set of descriptions of a representation $r \in \mathcal{B}^*$, and let $d \in \mathcal{D}_r$ be a description of r . We say that d is a *perfect description* of the representation r if there is no other description $d' \in \mathcal{D}_r$ such that $l(d') < l(d)$.

Recall that what we know about an entity e is conditional to the quality of the representation r used. If the representation r is wrong, we cannot reach a perfect knowledge about e even if we have found the perfect description d for r .

Notation 9.4. We denote by d_r^* the fact that the description d is a perfect description for the representation r .

The perfect description for a representation might not be unique, that is, there could be more than one optimal way to compute r .

Definition 9.5.5 Let \mathcal{D}_r be the set of descriptions of a representation $r \in \mathcal{B}^*$. We define the *set of perfect descriptions* for r , denoted by \mathcal{D}_r^* , as the subset of \mathcal{D}_r composed by the perfect descriptions of r .

Unfortunately, the set of perfect descriptions of a representation is in general not known and, as Proposition 9.5.1 shows, there exist no algorithm to compute it. In practice what we have to do is to use an approximation to estimate how far our current best description is from a perfect one, that is, how much we do not know about a particular representation for an entity (see Chapter 12).

Proposition 9.5.1 Let $r \in \mathcal{B}^*$ be a representation and d_r^* be a perfect description, then we have that $l(d_r^*) = K(r)$.

²The Occam's razor principle refers to the number of assumptions of an explanation, not to the length of the explanation itself.

Proof. Apply Definition 6.1.2 and the fact that we require that the Turing machines TM used in definitions $\langle TM, a \rangle$ must be prefix-free. ■

The actual length of a description $l(d)$ for a representation r is something that depends on the particular encoding of Turing machines used. The encoding method is given by the description function δ used. Fortunately, if we replace our description function by a different one, the length of perfect models do not change (up to an additive constant that does not depend on the representations themselves).

Corollary 9.5.2 Let $r \in \mathcal{B}^*$ be a representation, δ and $\dot{\delta}$ two different description functions, and d_r^* the perfect description of the representation r using δ and \dot{d}_r^* the perfect description using $\dot{\delta}$, then we have that $l(d_r^*) \leq l(\dot{d}_r^*) + c$ where c is a constant that does not depend on r .

Proof. Apply Theorem 9.5.1 and Theorem 6.1.1. ■

In general, in the theory of nescience we are not interested in computing the actual value of the nescience about an entity given a description and a representation. Instead what we are interested is in the ordering of the different possible pairs of descriptions and representations given their nescience. In this sense, the details of the particular universal Turing machine used in practice are not relevant³. For the rest of this book we will assume that δ is fixed to a reference universal Turing machine. For example, in Section 16.1.2 we use as universal Turing machine the lambda calculus. Alternatively, the reader could consider that all the theorems provided in this book that deal with the length of shortest models are valid up to an additive constant that does not depend on the topics themselves.

A remarkable consequence of Proposition 9.5.1 is that perfect descriptions must be incompressible, that is, *perfect knowledge implies randomness* (see Section 6.6).

Corollary 9.5.3 Let d_r^* be the perfect description of a representation r , then we have that $K(d_r^*) = l(d_r^*)$.

Proof. Having $K(d_r^*) < l(d_r^*)$ would be a contradiction with the fact that d_r^* is the shortest possible description of r . ■

The converse, in general, does not hold, since we could have a random description that it is not the shortest possible one, that is, a description d for a representation r such that $l(d) = K(d)$ but $l(d_r^*) < l(d)$.

■ Example 9.13 We could define a deep neural network with an input layer of one thousands nodes, ten hidden layers of fifty thousands nodes each, and an output layer of one thousand nodes, and then train the network to output a fixed string of one thousand 1's for any given input string. The Kolmogorov complexity of this neural network is much higher than the complexity of a string of one thousand 1's. ■

There is little value on descriptions that are longer than the representations they describe, that is, descriptions that do not compress the representations.

Definition 9.5.6 Let $r \in \mathcal{B}^*$ be a representation, and $d \in \mathcal{D}_r$ one of its descriptions. If $l(d) \geq l(r)$, we say that d is a *pleonastic description* of the representation r .

³Do not confuse the inner workings of the universal Turing machine that maps descriptions to representations, in which we are not interested, with the inner workings of the universal oracle Turing machine that maps representations to entities, in which we are interested, since this knowledge is critical to understand how things work.

■ **Example 9.14** Consider the set of all possible finite graphs. Since graphs are abstract mathematical objects, we need a way to represent them as strings, for example, by using a binary encoding of their adjacency matrices (see Section 2.5 for an introduction to graphs). The description $d = \langle TM, r \rangle$, where r is the representation of a graph and TM is a Turing machine that just halts, will be part of \mathcal{D}_r since $TM(r) = r$. We are concerned about the fact that this description may not be shortest possible description of r . ■

It might happen that there is no shortest possible description of a representation than the representation itself. This is the case when representations are random, or incompressible, strings. And, as we have seen in Section 6.6 the overwhelming majority of strings are incompressible. It is useless to do research about random representation, since it is not possible to find shorter models for that representation.

The concept of perfect description can be generalized from individual representations to entities. This generalization allow us to study the nature and properties of these entities.

Definition 9.5.7 Let \mathcal{D}_e be the set of descriptions of an entity $e \in \mathcal{E}$. We define the *set of perfect descriptions* of the entity e , denoted by \mathcal{D}_e^* as the subset of \mathcal{D}_e composed by perfect descriptions. The elements of \mathcal{D}_e^* are denoted by d_e^* .

If $d_e^* \in \mathcal{D}_e^*$ there must exists a $r \in \mathcal{R}_e^*$ such that $d_e^* \in \mathcal{D}_r^*$.

An interesting case is when all the descriptions that compose \mathcal{D}_e are pleonastics, that is, there is no model that it is shorter than the representation for all the possible representations of the entity. That would be the case if all the representations of the entity e are random strings. In this particular case, scientific research would be doomed, since it is not possible to find a suitable model for e . The fact that we can understand and make predictions about e will be limited by the length of the non-compressible representations of e .

9.6 Descriptions for Joint Representations

In Section 9.4 we introduced the concept of joint representation ts of two individual representations t and s . In this section we are interested in to study how the length of the perfect description of a joint representation relates to the length of the perfect descriptions of the individual representations.

The length of the perfect description of a joint representation is greater or equal than the length of the perfect description of the individual representations. That is, the more information we include in a representation, the longer will take to describe it.

Proposition 9.6.1 Let $t, s \in \mathcal{B}^*$ be two representations and m_t^* , m_s^* and m_{ts}^* the perfect descriptions of the representations t , s and the joint representation ts respectively. We have that $l(m_{ts}^*) \geq l(m_t^*)$ and $l(m_{ts}^*) \geq l(m_s^*)$.

Proof. The statement $l(m_{ts}^*) \geq l(m_t^*)$ is equivalent to $K(ts) \geq K(t)$. Then apply Proposition 6.3.3. ■

Intuitively, adding more information to a representation is a good thing if this information is relevant to describe the entity in which we are interested. But adding non-relevant information will make our model unnecessarily long. Recall that the process of joining representations can be used to concatenate two partial representations of the same entity, or to extend a representation with additional missing symbols.

If the selected representations partially overlap, we could take advantage of this redundancy to come up with a shorter join description than simply joining the individual descriptions. In the worst case, the perfect description of a joint representation would be the concatenation of the perfect descriptions of the individual representations.

Proposition 9.6.2 Let $t, s \in \mathcal{B}^*$ be two representations and m_t^* , m_s^* and m_{ts}^* the perfect descriptions of the representatons t , s and the joint representation ts respectively. We have that $l(m_{ts}^*) \leq l(m_t^*) + l(m_s^*)$.

Proof. The statement $l(m_{ts}^*) \leq l(m_t^*) + l(m_s^*)$ is equivalent to $K(ts) \leq K(t) + K(s)$, then apply Proposition 6.3.2. ■

An interpretation of Proposition 9.6.2 is that including redundant information in the representation of an entity is not a problem from the point of view of finding its shortest possible description. In practice, we have to use that representation that makes the process of scientific discovery (finding the best model) as easy as possible, even if this representation is unnecessarily long. In contrast, Proposition 9.6.1 claims that adding non-relevant symbols to a representation is something that has to be avoided.

Finally, next proposition proves that the order of the representations in the perfect description of a joint representation does not change its length.

Proposition 9.6.3 Let $t, s \in \mathcal{B}^*$ be two representations and m_{ts}^* and m_{st}^* the perfect descriptions of the joint representations ts and st respectively. We have that that $l(m_{ts}^*) = l(m_{st}^*)$.

Proof. The statement $l(m_{ts}^*) = l(m_{st}^*)$ is equivalent to $K(ts) = K(st)$, then apply Proposition 6.3.1. ■

Since joining representations is not a commutative operation, there is no way to guarantee that the strins ts and st encode the same entity. Note also that given the concatenated ts we cannot infer the original t and s , since they are not self-delimited strings.

Propositions 9.6.1, 9.6.2 and 9.6.3 can be generalized to any arbitrary, but finite, collection of representations t_1, t_2, \dots, t_n .

Proposition 9.6.4 Let $t_1, t_2, \dots, t_n \in \mathcal{B}^*$ be a finite collection of representations. Then, we have that:

- i $l(m_{t_1 t_2 \dots t_n}^*) \geq l(m_{t_i}^*) \forall 0 \leq i \leq n$,
- ii $l(m_{t_1 t_2 \dots t_n}^*) \leq l(m_{t_1}^*) + l(m_{t_2}^*) + \dots + l(m_{t_n}^*)$,
- iii $l(m_{t_1 \dots t_i \dots t_j \dots t_n}^*) = l(m_{t_1 \dots t_j \dots t_n}^*) + c \forall 0 \leq i \leq j \leq n$,
- iv $l(m_{t_1 \dots t_{n-1}}^*) \leq l(m_{t_1 \dots t_{n-1} t_n}^*)$.

Proof. Apply Propositions 9.6.1, 9.6.2 and 9.6.3 to individual pairs of topics i and j . ■

9.7 Conditional Descriptions

It is usually cumbersome to include all the information needed to reconstruct an entity in its description, since that would require very large strings for the majority of the entities. It is more convenient to assume some already existing background knowledge, and compute how much we do not know about an entity given that background. In this section we are going to study the concept of *conditional descriptions*, that is, computing a description given another description. Conditional descriptions also play a very important role in the discovery of new knowledge: if by conditioning a description to some prior knowledge we reduce significantly the inaccuracy of a model, that would mean this prior knowledge is relevant to understand the entity.

Definition 9.7.1 Let $r, d, s \in \mathcal{B}^*$ be strings. We say that the string $\langle d, s \rangle$ is a *valid conditional description* of the representation r given the string s , denoted by $d_{r|s}$, if $d = \langle TM, a \rangle$ is a description, and $TM(\langle a, s \rangle) = r$.

The conditional description $d_{r|s}$ is based on two strings a and s , that play very different roles. The string a is the input to the Turing machine TM , and it should contain the non-compressible part of the representation r . The string s should be the description, or representation, of another entity whose knowledge can help to understand the entity in which we are interested. For example, as we will see in Chapter 12, the string s is not taken into account when computing the surfeit of a conditional description.

Note that the conditional description $d_{r|s}$ does not belong to the set of valid description \mathcal{D} for the representation r , since s is required to compute the representation r , but it is not part of the description itself. A new definition is required to capture this new concept.

Definition 9.7.2 Let $r \in \mathcal{B}^*$ be a representation and $s \in \mathcal{B}^*$ an arbitrary string, we define the *set of conditional descriptions* of r given s , denoted by $\mathcal{D}_{r|s}$, as:

$$\mathcal{D}_{r|s} = \{d \in \mathcal{B}^*, d = \langle TM, a \rangle : TM(\langle a, s \rangle) = r\}.$$

For each representation $r \in \mathcal{B}^*$ there always exists a conditional description $d_{r|s}$ that describes r , as next proposition shows.

Proposition 9.7.1 Let $r \in \mathcal{B}^*$ be a representation and $s \in \mathcal{B}^*$ an arbitrary string. If $d \in \mathcal{D}_r$ then $d \in \mathcal{D}_{r|s}$.

Proof. We can use a conditional description $\langle \langle TM, a \rangle, s \rangle$ based on a Turing TM machine that given the input $\langle a, s \rangle$ safely ignores the s string. ■

The converse of Proposition 9.7.1 is not true. The fact that d is a conditional description ($d \in \mathcal{D}_{r|s}$) does not imply that d is also a description ($d \in \mathcal{D}_r$). We require that $TM(\langle a, s \rangle) = r$ but $TM(a) = r$ is not required, and it might not be the case.

We are interested in the concept of perfect conditional description. The perfect conditional description of a representation given a prior knowledge is the shortest possible string that allow us to fully reconstruct the representation assuming this prior knowledge.

Definition 9.7.3 Let $r \in \mathcal{B}^*$ be a representation, and let $d_{r|s}^*$ be the shortest possible description of r given the string s . We call $d_{r|s}^*$ the *perfect conditional description* of the representation r given the string s , or perfect conditional description of r given s for short.

Note that $d_{r|s}^*$ is a perfect description of the representation r conditional to the string s . That is, it might happen that the string s is not a perfect description itself, or it is a representation that contains non-relevant symbols. In that case, we would have reached a perfect knowledge with respect to the d part, but not for the s part, of the combined $\langle d, s \rangle$ string.

The length of perfect conditional description is equal or shorter than their unconditional counterparts. That is, assuming some already existing background knowledge could reduce the effort required to describe a representation.

Proposition 9.7.2 Let $r \in \mathcal{B}^*$ be a representation and $s \in \mathcal{B}^*$ an arbitrary string. We have that $l(d_{r|s}^*) \leq l(d_r^*)$.

Proof. The statement $l(d_{r|s}^*) \leq l(d_r^*)$ is equivalent to $K(r | s) \leq K(r)$, then apply Proposition 6.4.3. ■

Next proposition shows the relation between the lengths of descriptions, join descriptions and conditional descriptions.

Proposition 9.7.3 Let $r, s \in \mathcal{B}^*$ two different representations. We have that:

$$l(d_{r|s}^*) \leq l(d_r^*) \leq l(d_{rs}^*)$$

Proof. The statement $l(d_{r|s}^*) \leq l(d_r^*) \leq l(d_{rs}^*)$ is equivalent to $K(r|s) \leq K(r)$ and $K(r) \leq K(rs)$, then apply Proposition 6.4.4. ■

As it was the case of joint descriptions, the concept of conditional description can be extended to finite collections of representations.

Definition 9.7.4 Let $r, d, s_1, s_2, \dots, s_n \in \mathcal{B}^*$ be strings. We say that the string $\langle d, s_1, s_2, \dots, s_n \rangle$ is a *valid conditional description* of the representation r given the strings s_1, s_2, \dots, s_n , denoted by $d | s_1, s_2, \dots, s_n$, if $d = \langle TM, a \rangle$ is a description, and $TM(\langle a, s_1, s_2, \dots, s_n \rangle) = r$.

In the next definition we provide the generalization of the concept of perfect conditional descriptions.

Definition 9.7.5 Let $r \in \mathcal{B}^*$ be a representation, and let $d_{r|s_1, s_2, \dots, s_n}^*$ be the shortest possible description of r given the strings s_1, s_2, \dots, s_n . We call $d_{r|s_1, s_2, \dots, s_n}^*$ the *perfect conditional description* of the representation r given the string s_1, s_2, \dots, s_n , or perfect conditional description of r given s_1, s_2, \dots, s_n for short.

Next proposition generalizes Propositions 9.7.2 and 9.7.3 to any arbitrary, but finite, collection of strings s_1, s_2, \dots, s_n . Moreover, the proposition shows that the more background knowledge we assume for a representation, the shorter is the perfect description for that representation.

Proposition 9.7.4 Let $r, s_1, s_2, \dots, s_n \in \mathcal{B}^*$ be a finite collection of strings. Then, we have that:

$$l(d_{r|s_1, s_2, \dots, s_n}^*) \leq l(d_r^*) \leq l(d_{r, s_1, s_2, \dots, s_n}^*)$$

Proof. Apply Propositions 9.7.2 and 9.7.3 to individual pairs of representations i and j . ■

The following proposition generalizes the idea that assuming more background knowledge to a description cannot increase its length.

Proposition 9.7.5 Let $r, s_1, s_2, \dots, s_n, s_{n+1} \in \mathcal{B}^*$ be a finite collection of strings. Then, we have that:

$$l(d_{r|s_1, s_2, \dots, s_n, s_{n+1}}^*) \leq l(d_{r|s_1, s_2, \dots, s_n}^*)$$

Proof. Apply Propositions 9.7.2 and 9.7.3 to individual pairs of representations i and j . ■

9.8 Research Areas

Entities can be grouped into research areas. The concept of area is useful as long as all the entities included in the area are related to a common knowledge subdomain, or share a common property. The particular details of the grouping criteria depend on the practical applications in which the theory of nescience is being used.

Definition 9.8.1 Given a set of entities \mathcal{E} , we define a *research area* \mathcal{A} as a subset of entities $\mathcal{A} \subset \mathcal{E}$.

If we want to know how much we do not know about a research area, first we have to provide a representation for that area. In general areas are infinite, but the number of known representations is finite, and so, we can only describe the areas with respect to our current knowledge.

Definition 9.8.2 Let $\mathcal{A} \subset \mathcal{E}$ be an area. We define the *known subset of the area \mathcal{A}* , denoted by $\hat{\mathcal{A}}$, as the set composed by those entities $e_1, e_2, \dots, e_n \in A$ for which at least one non-pleonastic description is known.

We have to distinguish between the knowable subset of \mathcal{A} , composed by those entities for which there exists a representation, and the known subset of \mathcal{A} , composed by those entities for which we know a non-pleonastic description, that is, those entities for which somebody has already done some research about them. Of course, the set of known entities is a subset of the set of knowable entities.

As our understanding of a research area changes, the number of entities included in its known subset changes as well. The properties of areas studied in this book will be always relative to our current knowledge.

Definition 9.8.3 Let $\mathcal{A} \subset \mathcal{E}$ be an area with known subset $\hat{\mathcal{A}} = \{e_1, e_2, \dots, e_n\}$, and let $R = \{r_1, r_2, \dots, r_n\}$ a set of representations, such that $r_i \in \mathcal{R}_{e_i}$. We call $R_{\hat{\mathcal{A}}}$ a *representation of the area \mathcal{A}* given the known subset $\hat{\mathcal{A}}$, abbreviated as *representation of A* .

In the same way, we can introduce the concept of description of an area.

Definition 9.8.4 Let $R_{\hat{\mathcal{A}}} = \{r_1, r_2, \dots, r_n\}$ be the representation of an area \mathcal{A} . We call a *description of the area \mathcal{A}* given the known subset $\hat{\mathcal{A}}$, abbreviated as *description of A* , and denoted by $d_{\hat{\mathcal{A}}}$, to any string in the form $\langle TM, a \rangle$ such that $TM(a) = \langle r_1, r_2, \dots, r_n \rangle$.

We can also define the set of descriptions of an area.

Definition 9.8.5 Let $R_{\hat{\mathcal{A}}} = \{r_1, r_2, \dots, r_n\}$ be the representation of an area \mathcal{A} . We define the set of *descriptions for $R_{\hat{\mathcal{A}}}$* , denoted by $\mathcal{D}_{R_{\hat{\mathcal{A}}}}$, as:

$$\mathcal{D}_{R_{\hat{\mathcal{A}}}} = \{d \in \mathcal{D} : TM(a) = \langle r_1, r_2, \dots, r_n \rangle\}.$$

Finally, we are interested in the perfect model for a research area, that is, the shortest possible string that fully describes its known subset. According to Definition 9.5.6, if we are aware of the existence of a entity $e \in A$, that entity should be part of $\hat{\mathcal{A}}$, even in the case we have not started yet to do research about that particular topic.

Definition 9.8.6 Let $\mathcal{A} \subset \mathcal{E}$ be an area with known subset $\hat{\mathcal{A}}$, and let $d_{\hat{\mathcal{A}}}^* \in \mathcal{M}_{\hat{\mathcal{A}}}$ be the shortest possible description of A . We call $d_{\hat{\mathcal{A}}}^*$ the *perfect description of the area A* given the known subset $\hat{\mathcal{A}}$, abbreviated as *perfect description of A* .

Next proposition shows the relation between the description of an area, and the descriptions of the entities that compose the known subset of that area. In general, the models for an area are different from the collection of models of the individual topics.

Proposition 9.8.1 Let $\mathcal{A} \subset \mathcal{E}$ be an area with known subset $\hat{\mathcal{A}} = \{e_1, e_2, \dots, e_n\}$, then we have that $l(d_{\hat{\mathcal{A}}}^*) \leq l(d_{e_1}^*) + l(d_{e_2}^*) + \dots + l(d_{e_n}^*)$.

Proof. Apply Proposition 9.6.4-ii. ■

Also, as it was proved in Proposition 9.6.4, the order in which the representations are listed in the description of an area is not relevant when dealing with the perfect model for that area.

Areas can overlap, that is, given two areas A and B it might happen that $A \cap B \neq \emptyset$. Moreover, areas can be subsets of other areas, creating an hierarchy of areas. We are interested in the length of perfect models of areas in relation to the length of perfect models of other areas.

Proposition 9.8.2 Let $A, B \subset \mathcal{E}$ be two areas such that $A \subset B$, and let \hat{A} and \hat{B} be their know subsets respectively, then we have that $l(d_{\hat{A}}^*) \leq l(d_{\hat{B}}^*)$.

Proof. TODO ■

Next proposition shows how the length of the shortest possible description of areas relate to the union and intersection of such areas.

Proposition 9.8.3 Let $A, B \subset \mathcal{E}$ be two areas with know subsets \hat{A} and \hat{B} respectively, then we have that $l(d_{\hat{A} \cup \hat{B}}^*) = l(d_{\hat{A}}^*) + l(d_{\hat{B}}^*) - l(d_{\hat{A} \cap \hat{B}}^*)$.

Proof. TODO ■

A consequence of Proposition is that $l(d_{\hat{A} \cup \hat{B}}^*) \leq l(d_{\hat{A}}^*) + l(d_{\hat{B}}^*)$, that is, when we combines two different research areas, how much we do not know about these areas decreases.

In the same way we introduced a chain rule for entropy in Proposition 5.4.5, we can provide a chain rule for the shortest length for a description of a research area.

Proposition 9.8.4 Let $A, B \subset \mathcal{E}$ be two areas with know subsets \hat{A} and \hat{B} , then we have that $l(d_{\hat{A} \cup \hat{B}}^*) = l(d_{\hat{A}}^*) + l(d_{\hat{B} \setminus \hat{A}}^*)$.

Proof. TODO ■

9.9 References

TODO: Review this section.

For more information about Russell's paradox, Cantor theorem and universal sets refer, for example, to [[jech2013set](#)]. The idea of using a function to assigns to each symbol and well-formed formula of some formal language a unique natural number (Gödel number) was introduced by Kurt Gödel for the proof of his incompleteness theorems [[godel1931formal](#)]. A detailed description of the Berry paradox from the point of view of computability can be found at [[chaitin1995berry](#)]. For a detailed account of the implications of Kolmogorov complexity being true up to a constant, please refer to [[li2013introduction](#)]. That oracle machines are not mechanical was stated by Turing when he introduced the concept of oracle machine in [[turing1939systems](#)].

- Intro to ontology: the things we can know about
- Intro to epistemology: the problem of representation
- What it is a representtaion: reference to oxford philosophy article
- Reference to the concept of what it is research topic
- The problem of representation in Kolmogorov complexity
- Godel numbering
- single, unique, physical world -> what it is thing called
- Cantor theorem
- Rusell paradox
- Oracle Turing machine
- ptolomeo and tycho

universal oracle machine

Should be require the oracles to be minimal?



10. Miscoding

*All great work is the fruit of patience and perseverance,
combined with tenacious concentration on a subject
over a period of months or years.*

Santiago Ramón y Cajal

In many areas of science, the things we want to study - the set of entities or \mathcal{E} - often include abstract ideas or complicated things that are tough to investigate directly. For example, these could be concepts or phenomena that are difficult to describe or model. To understand such entities, we need to use a kind of roundabout method. As we have seen in the previous chapter, the theory of nescience proposes we use representations (sequences of symbols) instead of directly interacting with the actual entities. But this comes with its own issues: because our understanding of the elements of \mathcal{E} is incomplete (otherwise we would not need to keep researching), the representations we use are usually imperfect - they are not as comprehensive or as precise as we would ideally want them to be. These imperfections can cause big problems because inaccuracies in how we represent something can lead to major errors in the model we use to describe it, which can then mess up our interpretation and understanding. It is really important to be aware of this potential source of error and get a handle on its implications.

To help with this, we introduce *miscoding*, a measure we use to size up the errors that come from misrepresenting or inaccurately encoding these things. We define miscoding based on the length of the shortest computer program that can correct a wrong representation to make it right. Basically, miscoding measures the 'effort' (measured by the length of the program) needed to fix a wrong representation. But, there is a catch: putting this idea into practice is not straightforward, because the perfect representations of things are unknown, so we can't measure the gap or difference between our current representation and a perfect one. Even so, we suggest a theoretical approach based on the oracle machine (an abstract idea) that is used to define the set \mathcal{E} . This oracle machine can recognize valid representations for all entities. But, we need to be aware of some limitations about the questions we can ask the oracle. For example, we can't ask the oracle about a specific entity for which we do not have a valid representation.

Getting a better grasp on scientific representation and the challenges in reducing miscoding could really push forward scientific research, helping us create better, more thorough models of the natural world. In this chapter, we are going to properly introduce the idea of miscoding and look into its various characteristics. We will examine how miscoding behaves when it comes to combined representations and discuss methods to reduce miscoding. We will delve into the idea that there are strings that do not represent any entity. And we will dig into why miscoding is important in different research areas and talk about its potential to open up new lines of inquiry.

10.1 Miscoding

In an ideal situation, we would ask the oracle to measure how far is a particular string r from perfectly encoding an entity e . This creates a complication because we can only inform the oracle about entity e through the use of a valid representation from the set \mathcal{R}_e^* . However, we typically do not have access to these perfect representations, and thus, cannot use them in our query. To address this impediment, we propose an alternative approach. Instead of asking the oracle about the discrepancy between our representation r and a valid representation of the target entity e , we propose to ask the oracle about the smallest distance between our current representation r and all the valid representations of all entities encompassed within the set \mathcal{E} , that is, the elements of the set $\mathcal{R}_{\mathcal{E}}^*$. As we have discussed in detail in Section 9.3, this alternative form of query is one that the oracle can theoretically handle and provide answers to.

Definition 10.1.1 — Miscoding. Let $r \in \mathcal{B}^*$ be a representation. The *miscoding* of r , denoted by $\mu(r)$, is defined as:

$$\mu(r) = \min_{r_e^* \in \mathcal{R}_{\mathcal{E}}^*} \frac{\max\{K(r | r_e^*), K(r_e^* | r)\}}{\max\{K(r), K(r_e^*)\}}$$

where \min^o denotes that the minimum has to be computed by an oracle.

Intuitively, our ignorance regarding an entity corresponds to a higher miscoding value of our current representation. Conversely, gaining a deeper understanding of an entity should allow for an encoding that is closer to a valid representation.

Miscoding is computed through a bidirectional approach: we require the oracle to compute the length of the shortest computer program capable of generating the string r_e^* given our representation r , and vice versa – that is, to compute the length of the shortest computer program capable of generating r given the string r_e^* . As expected, a high-quality representation should contain all the necessary information to reconstruct an entity, devoid of any erroneous or superfluous data. Miscoding involves the inclusion of only relevant symbols, while surfeit (as discussed in Chapter 12) pertains to the inclusion of only necessary symbols.

Our definition of miscoding employs a relative measure as opposed to an absolute one. This choice allows us to compare the miscoding of various encodings for the same entity, as well as to compare the miscoding across different entities.

The miscoding value of a representation r consistently falls between 0 and 1.

Proposition 10.1.1 For all $r \in \mathcal{B}^*$, it holds that $0 \leq \mu(r) \leq 1$.

Proof. This is ensured by $0 \leq \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \leq 1$ for all $x, y \in \mathcal{B}^*$ as stated in Proposition 6.5.3. ■

Miscoding is zero if, and only if, the representation r is a valid representation of a particular entity e .

Proposition 10.1.2 Given a representation $r \in \mathcal{B}^*$, we have $\mu(r) = 0$ if, and only if, $r \in \mathcal{R}_{\mathcal{E}}^*$.

Proof. If $r \in \mathcal{R}_e^*$, there exists an entity $e \in \mathcal{E}$ such that $r = r_e^*$, thus, $K(r | r_e^*) = 0$ and $\mu(r) = 0$. If $\mu(r) = 0$, there must exist an $r_e^* \in \mathcal{R}_e^*$ such that $K(r | r_e^*) = 0$, implying that $r = r_e^*$ and $r \in \mathcal{R}_e^*$. ■

According to Proposition 10.1.2, if the miscoding of r equals 0, it can be concluded that r perfectly encodes an entity e . The challenge lies in determining the exact entity encoded by r . Although scientific intuition may guide guesses regarding the encoded entity, it is not mathematically possible to substantiate these conjectures. Furthermore, with the progress of research, perceptions of the encoded entity's nature may evolve over time (see Example 9.10).

It has been observed that an entity $e \in \mathcal{E}$ may have multiple valid representations, as denoted by the set \mathcal{R}_e^* . Fortuitously, miscoding is a value that remains unaffected by the chosen representation (refer to the issue of style in Section 8.2).

Proposition 10.1.3 For any valid representation $r_e^* \in \mathcal{R}_e^*$, it holds that $\mu(r_e^*) \leq \mu(r)$ for all $r \in \mathcal{B}^*$.

Proof. This follows from $\mu(r_e^*) = 0$ and $\mu(r) \geq 0$ for all $r \in \mathcal{B}^*$. ■

For a given entity e , all valid representations encompassed by \mathcal{R}_e^* exhibit an equivalent degree of adequacy in terms of miscoding, given that each maintains a miscoding value of 0. Pragmatically, the representation that facilitates the most efficient accumulation of new knowledge about the entity in question should be selected, specifically the one that enables the derivation of models explicating the entity.

10.2 Joint Miscoding

Section 9.4 introduced the notion of 'joint representation', which arises when two representations, $s, t \in \mathcal{B}^*$, are concatenated to form a new string st . This section delves into the properties of miscoding intrinsic to these joint representations.

Given that the joint representation st qualifies as a representation, it is not necessary to introduce a new definition of miscoding to capture this concept. The miscoding of a joint representation is given by:

$$\mu(st) = \min_{r_e^* \in \mathcal{R}_e^*} \frac{\max\{K(st | r_e^*), K(r_e^* | st)\}}{\max\{K(st), K(r_e^*)\}}$$

In alignment with Proposition 10.1.1, the miscoding of a joint representation is a value between 0 and 1, expressed as $0 \leq \mu(st) \leq 1$.

It is worth noting that the process of supplementing an incomplete representation — a representation with a positive miscoding — with additional symbols does not unequivocally guarantee a reduction in miscoding. This is because the appended symbols may introduce errors. Conversely, it is also not certain that the miscoding will increase, as the integration of pertinent symbols can potentially decrease the miscoding of an incomplete representation. From a formal point of view, given two arbitrary representations $s, t \in \mathcal{B}^*$, there is no guarantee that any of the following relationships will consistently hold: $\mu(ts) \geq \mu(t)$, $\mu(ts) \leq \mu(t)$, $\mu(ts) \geq \mu(s)$, nor $\mu(ts) \leq \mu(s)$.

■ **Example 10.1** Consider the text r of a biology research article that succinctly and accurately describes a newly discovered specimen, resulting in a low miscoding value $\mu(r)$. However, when additional text s from a completely unrelated article on a second specimen from a different species is appended, the new concatenated representation rs muddles the original focus. This inclusion of unrelated content increases the miscoding value $\mu(rs) = 0.6$, proving the case where $\mu(rs) \geq \mu(r)$. The added text, though scientific, dilutes the clarity and precision of the original article, causing a rise in miscoding.

Conversely, in a second scenario, another biological research article r' has a relatively high miscoding value of $\mu(r')$, stemming from an incomplete presentation of the specimen. However, the

concatenation of an additional text s' , rich with the previously omitted essential details, ameliorates the article. The concatenated version ts thus sees a significant reduction in miscoding. This exemplifies the case of $\mu(ts) \leq \mu(t)$ and underscores the premise that the enrichment of an incomplete representation with pertinent, focused information can lead to a more accurate, reliable, and lower miscoding value. ■

Furthermore, it should not be assumed that the miscoding of a joint representation is less than the sum of the miscoding of the individual representations, i.e., $\mu(st) \leq \mu(s) + \mu(t)$. This holds true even when both representations t and s encode the same entity, as the joint representation ts might encode an entirely distinct entity compared to t and s .

■ **Example 10.2** In a medical research context, representation r describes a new drug compound A for treating a specific type of cancer, while representation s outlines a genetic mutation B associated with that cancer type. Both r and s have low miscoding values, denoting accurate, focused content. However, when concatenated, the combined representation rs unintentionally suggests a third entity C , implying a causal relationship between A and B . This unintended implication stems from the mixture of distinct information, leading to a higher miscoding value. In this instance, each of r , s , and rs essentially represents different research entities. r and s are clear in their individual contexts, but rs is ambiguous, exemplifying a case where concatenated information can inadvertently create a representation of an entirely different, unintended entity, thus increasing miscoding. ■

Given the non-commutative nature of joining two representations, it is not guaranteed that $\mu(ts) = \mu(st)$. This is because the composite strings ts and st could potentially encode two different entities. This property hold true even when we restrict our focus to the valid representations of a specific entity e . For instance, if $r_e^*, s_e^* \in \mathcal{R}_e^*$ denote two valid representations of the entity e , there is no guarantee that the concatenated string $r_e^*s_e^*$ will maintain its status as a representation of e .

■ **Example 10.3** In environmental science, representation r thoroughly examines microplastics, their nature, and detrimental impacts on marine ecosystems, while representation s outlines specific technologies and methodologies for detecting these pollutants in the water. The rs concatenation offers a logical narrative from the identification of microplastics and their effects to the methodologies of detection, forming a coherent entity that serves as a comprehensive resource on the pollutant, its impacts, and detection methods, and which is not fully understood. Conversely, sr starts with the technical and methodological aspects of detecting microplastics and then moves to describe the pollutants and their effects, forming another clear entity that can serve as a methodological guide with a detailed context of application, with low miscoding. In this particular case we have that $\mu(rs) > \mu(sr)$ ■

The concept of miscoding can be generalized to apply to joint representations of any finite collection of representations. If $r_1, r_2, \dots, r_n \in \mathcal{B}^*$ comprise a finite collection of representations, the miscoding of the joint representation $r_1r_2\dots r_n$, can be expressed as:

$$\mu(r_1r_2\dots r_n) = \min_{r_e^* \in \mathcal{R}_e^*} \frac{\max\{K(r_1r_2\dots r_n | r_e^*), K(r_e^* | r_1r_2\dots r_n)\}}{\max\{K(r_1r_2\dots r_n), K(r_e^*)\}}$$

Consistent with the case of joining two representations, we maintain that $0 \leq \mu(r_1, r_2, \dots, r_n) \leq 1$. It cannot be assured that $\mu(r_1, r_2, \dots, r_n) \leq \mu(r_i)$ or $\mu(r_1, r_2, \dots, r_n) \geq \mu(r_i)$ for $i = 1, \dots, n$. Additionally, it is not necessarily true that $\mu(r_1, r_2, \dots, r_n) \leq \mu(r_1) + \dots + \mu(r_n)$. Lastly, it is not possible to make any assertions regarding the miscoding of a permutation of the representations incorporated in the joint representation compared to the original miscoding.

10.3 Decreasing Miscoding

A valid representation of an entity refers to a string that encapsulates all the vital information needed by the oracle to reproduce the given entity, and solely that information. Conversely, an invalid representation—signified by a miscoding exceeding zero—may stem from either a lack of critical information, the presence of incorrect symbols, or the inclusion of irrelevant symbols. To decrease the miscoding of a representation, one can supplement missing information or eliminate incorrect or non-pertinent symbols. Determining in advance whether any information is absent or if certain symbols require removal is not possible. Nonetheless, as the ensuing theorem demonstrates, one of these scenarios must inevitably be true.

Theorem 10.3.1 Let $r \in \mathcal{B}^*$ be a representation such that $\mu(r) > 0$, then one or both of the following conditions must hold:

- (i) There exists a $s \in \mathcal{B}^*$ such that $\mu(rs) < \mu(r)$ or $\mu(sr) < \mu(r)$,
- (ii) There exists an $s \in \mathcal{B}^*$ in the form $r = \alpha s \beta$ with $\alpha, \beta \in \mathcal{B}^*$ such that $\mu(s) < \mu(r)$.

Proof. To be completed Assuming that $\mu(r) > 0$, we find that

$$\min_{r_e^* \in \mathcal{R}_e^*} \frac{\max\{K(r | r_e^*), K(r_e^* | r)\}}{\max\{K(r), K(r_e^*)\}} > 0$$

Let $r_e^* = \arg \min(\mu(r))$. Hence, $\max\{K(r | r_e^*), K(r_e^* | r)\} > 0$. If $K(r | r_e^*) > 0$, it indicates that r includes non-relevant symbols. Conversely, if $K(r_e^* | r) > 0$, it implies that r omits some relevant symbols. ■

■ **Example 10.4** Suppose that the research entity e in which we are interested is the causes of lung cancer. In order to understand this entity, we have measured a collection of risk factors in a random sample of the population (smoking, exercise, diet, age, etc.). However, due to a problem with the sampling procedure, all the samples correspond to a subset of the population, for example, males. This dataset would be a representation s for our entity e , but a very bad one, since it is strongly biased. If we have a second representation, corresponding the sample data of females t , the joint representation st will be a better one than any of them, s or t , isolated. ■

Next example describes a case in which removing some symbols from a representation can decrease its miscoding.

■ **Example 10.5** A historian compiles representation r , aiming to narrate the events surrounding the signing of a pivotal treaty. However, the inclusion of speculative statements and personal interpretations about the involved parties' intentions, not backed by primary sources, leads to a high miscoding value. The document, while rich in information, is clouded by content that is erroneous, making it an invalid representation of the historical event. By removing these speculative and unverified segments, a refined version r' emerges, offering a concise, factual account based solely on verifiable data and primary sources. This refinement leads to a decrease in the miscoding value, transforming r' into a more valid representation that accurately reflects the historical event without the noise of uncorroborated interpretations. ■

10.4 Targetless Representations

In the most extreme scenario, the miscoding value could escalate to its maximum limit of 1. This situation transpires when our prevailing representation r does not contain a single symbol that corresponds to the encoding of any entity within the set of entities \mathcal{E} . In this case, r would be a targetless representation, an abstract construct without a concrete or meaningful interpretation. We

could assign randomly the targetless representations to an entity, but that would violate the highly desirable property of subjective reasoning.

Next proposition formalizes this idea.

Proposition 10.4.1 Given a representation $r \in \mathcal{B}^*$, the miscoding $\mu(r)$ reaches its maximum value of 1 if and only if there is no symbol in r that contributes to the encoding of any of the entities of \mathcal{E} .

Proof. Assume that r does not contain any symbol that contributes to the encoding of e . Let r_e^* be any valid representation of e . Then the shortest program capable of generating r_e^* from r would need to append every symbol in r_e^* to r , effectively resulting in r_e^* . Hence, $K(r_e^*|r) = K(r_e^*)$, indicating no symbolic overlap between r and r_e^* .

Following the definition of miscoding, we have $\mu(r) = \max K(r|r_e^*), K(r_e^*|r) / \max K(r), K(r_e^*)$. Because $K(r|r_e^*) = K(r)$ and $K(r_e^*|r) = K(r_e^*)$, we get $\mu(r) = 1$, assuming that $K(r) \neq 0$.

The converse also holds. If $\mu(r) = 1$, this means that there's no information in r that assists in generating r_e^* or vice versa, which implies that r does not contain any symbol that contributes to the encoding of e . ■

For every finite or countable infinite set of entities \mathcal{E} , there exists a infinite number of targetless representation. We will prove this property using a constructive method. That is, we will prove that there exists at least one targetless representation and define a method to construct more targetless representations given the original one.

Proposition 10.4.2 For every finite or countably infinite set of entities \mathcal{E} , there exists an uncountably infinite set of targetless representations.

Proof. Let \mathcal{R} denote the set of all possible representations for entities in \mathcal{E} , such that for each entity $e \in \mathcal{E}$, there exists a unique representation $r \in \mathcal{R}$ and vice versa. Since \mathcal{E} is finite or countably infinite, so is \mathcal{R} .

Consider the set \mathcal{B}^* of all possible binary strings of finite length. This set is uncountably infinite, because it includes all possible sequences of two symbols ('0' and '1') of finite length.

Since \mathcal{R} is countable (either finite or countably infinite), and \mathcal{B}^* is uncountably infinite, there must exist some strings in \mathcal{B}^* that are not in \mathcal{R} . We can select any such string and call it r_0 . By definition, r_0 is a targetless representation, as it does not correspond to any entity in \mathcal{E} .

Given a targetless representation r_i , we can construct a new targetless representation r_{i+1} by appending either a '0' or a '1' to r_i . Since r_i does not correspond to any entity in \mathcal{E} , r_{i+1} , being a lengthened version of r_i , also cannot correspond to any entity in \mathcal{E} . Therefore, r_{i+1} is also a targetless representation.

By repeating this process, we can construct an infinite sequence of targetless representations $\{r_0, r_1, r_2, \dots\}$, demonstrating that there exists an uncountably infinite set of targetless representations for any finite or countably infinite set of entities \mathcal{E} . ■

What it is a targetless representation for an oracle is not necesarily a target representation for another oracle, as next example shows.

■ **Example 10.6** Imagine two machines, A and B, each controlling a robotic arm adept at crafting nuts and bolts. Machine A operates on a low-level assembly language, while Machine B utilizes a more sophisticated high-level programming language. Consequently, a particular set of instructions that proves inadequate and fails to yield a finished product with Machine A could be perfectly executed by Machine B, resulting in a completed bolt. ■

The normalized compression distance between a targetless representation and the closest non-targetless representation might be less than one, meaning that the target-less representation contains

some information about the non-targetless one. However, this is not sufficient to guarantee scientific progress, given the size of the Turing machine needed to benefit from this knowledge.

10.5 Miscoding of Areas

The concept of miscoding can be extended to research areas in order to quantitatively measure the amount of effort required to fix an inaccurate representation of an area. Unfortunately, as we have discussed in Section 9.8, there is no way to check if the strings included in a n -fold representation correspond to entities of that area, or to avoid the fact there may be instances where some of these strings represent the same entity.

Let's consider the strings r_1, r_2, \dots, r_n , where each $r_i \in \mathcal{B}^*$ for $i = 1, 2, \dots, n$. Recall that when we write $r_1 r_2 \dots r_n$, we refer to the concatenation of these strings. This operation could potentially meld the individual strings in such a way that they cannot be separated back into their original components. Conversely, by $\langle r_1, r_2, \dots, r_n \rangle$ we denote a re-encoding of the individual strings r_i into a single string, but with the provision that it retains the ability to decompose into the original strings. Moreover, the joint representation $r_1 r_2 \dots r_n$ encodes a single entity, while the n -folded representation $\langle r_1, r_2, \dots, r_n \rangle$ might encode up to n distinct entities.

The following definition extends the concept of miscoding to n -fold representations.

Definition 10.5.1 Let $R = (r_1, r_2, \dots, r_n) \in \mathcal{B}^* \times \mathcal{B}^* \times \dots \times \mathcal{B}^*$ be an n -fold representation. We define the *misCoding* of R , denoted by $\mu(R)$, as:

$$\mu(R) = \min_{(r_{e_1}^*, \dots, r_{e_n}^*) \in \mathcal{R}_{\mathcal{E}}^* \times \dots \times \mathcal{R}_{\mathcal{E}}^*} \frac{\max\{K(\langle r_1, \dots, r_n \rangle | \langle r_{e_1}^*, \dots, r_{e_n}^* \rangle), K(\langle r_{e_1}^*, \dots, r_{e_n}^* \rangle | \langle r_1, \dots, r_n \rangle)\}}{\max\{K(\langle r_1, \dots, r_n \rangle), K(\langle r_{e_1}^*, \dots, r_{e_n}^* \rangle)\}}$$

where \min^o denotes that the minimum has to be computed by an oracle.

The miscoding of the representation of an area falls within the range of 0 and 1, as illustrated by the following proposition.

Proposition 10.5.1 For all known subsets $R = (r_1, r_2, \dots, r_n) \in \mathcal{B}^* \times \mathcal{B}^* \times \dots \times \mathcal{B}^*$, we have that $0 \leq \mu(R) \leq 1$.

Proof. Given that $\langle r_1, r_2, \dots, r_n \rangle$ is a string and Proposition 6.5.3. ■

By extending the concept of miscoding to encompass areas, we can quantitatively evaluate the quality of representations for specific subsets of entities. This mathematical framework offers a robust tool to assess and rectify inaccuracies in both individual entities and broader research areas.

References

Misrepresentation or inaccuracies in scientific representation carry substantial implications for scientific discovery, technological advancement, policymaking, among others. However, no book or paper explicitly addresses the subject of "incorrect representations in science," from the perspective adopted in this book, in which we dissect the issue of scientific representation into two complementary subproblems: representation of entities and description of representations. Despite the absence of a direct focus, a variety of researchers have ventured into this domain indirectly. Their discussions often touch on concerns such as scientific fraud, the replication crisis, and the usage of incorrect models, to name a few.



11. Inaccuracy

A little inaccuracy sometimes saves tons of explanations.

Saki

In Section 9.5, we introduced the idea of a description, or a model, of an entity as a computer program. When this program is executed, it reproduces one of the representations encoding the entity in question. More precisely, a description d for a representation r of an entity e is a Turing machine that produces the string r when a universal Turing machine δ interprets it. However, given our typically incomplete understanding of the entity e being studied, the actual output of the description $\delta(d)$, denoted as r' , will be similar, but not identical to r . In this chapter, we will explore the error brought about by flawed models—specifically, how closely r' approximates the original string r . We denote this form of error as the inaccuracy of the description d .

Inaccuracy serves as the second metric in assessing our understanding of a research entity. The underlying idea is that the more accurate our model, the better our understanding of the entity. Formally, we calculate the inaccuracy of a description d as the normalized information distance between the original representation r and the output representation r' generated by our description d . That is, inaccuracy is quantified as the length of the smallest computer program capable of correcting the erroneous output of our model.

Inaccuracy, serving as the second gauge to measure our comprehension of a research entity, is based on the principle that the more precise our model, the better our grasp of the entity. Formally, the inaccuracy of a description d is computed as the normalized information distance between the original representation r and the output representation r' generated by the description d . Thus, inaccuracy is assessed as the extent of the smallest computer program that can rectify the incorrect output of our model.

Inaccuracy evaluates how well the output of our description aligns with the selected representation encoding the entity. However, this representation could be flawed itself, as discussed in the preceding chapter. Inaccuracy focuses solely on the description d , neglecting the potential miscoding within the representation r . Furthermore, even though it doesn't require an oracle,

inaccuracy cannot be calculated for every case, so it needs to be estimated in practical situations, as we will explore in Part III of this book.

In this chapter, we will formally define inaccuracy and examine its characteristics. We will also scrutinize how inaccuracy varies when a conditional description of a representation is used as opposed to an unconditional one. Finally, we will extend the concept of inaccuracy from individual entities to entire research areas.

This study of inaccuracy is not merely academic but has practical implications as well. Accurate models are crucial to many fields, from predicting climate change to designing artificial intelligence systems. Therefore, understanding and quantifying inaccuracy can lead to improvements in these models, allowing us to make better predictions and decisions.

11.1 Inaccuracy

In the process of studying an entity $e \in \mathcal{E}$ through a representation $r \in \mathcal{R}_e$, we may encounter situations where our proposed description d fails to provide an accurate depiction for r . In other words, $d \notin \mathcal{D}_r$ (refer to Definition 9.5.2). Under such circumstances, when the universal Turing machine δ receives d as an input, it will yield a string r' that diverges from the original string r . On an intuitive level, one could infer that d serves as an inaccurate description of the entity e . However, given that descriptions convey entities indirectly via representations, our formal interpretation of inaccuracy must be predicated upon the representations employed, rather than the original entities. We must also acknowledge the possibility of representations being inherently flawed, an issue previously addressed through the notion of miscoding. With these considerations in mind, we propose the following definition for the concept of an inaccurate description.

Definition 11.1.1 Let $r \in \mathcal{B}^*$ represents a representation, and $d \in \mathcal{D}$ a description, where $d = \langle TM, a \rangle$. If the outcome of $TM(a)$ is a string r' , such that $r \neq r'$, we designate d as an *inaccurate* description for r .

Our proposed description d may not fall within the spectrum of valid descriptions \mathcal{D}_r for r (an instance of positive inaccuracy), and the representation r may not be included in the valid \mathcal{R}_e^* representations for the entity e (indicative of positive miscoding).

In instances where our description proves inaccurate, we aspire to quantify the extent of this inaccuracy. In the context of computational machines, an intuitive approach to defining this measure would involve calculating the difficulty in converting the incorrect representation r' —produced by running d through the universal Turing machine—into the original representation r . In essence, this involves the computation of the normalized information distance between r' and r .

Definition 11.1.2 — Inaccuracy. Let us consider $r \in \mathcal{B}^*$ as a representation, and $d \in \mathcal{D}$ as a description, where $d = \langle TM, a \rangle$. We then define the *inaccuracy* of the description d with respect to the representation r , denoted as $\iota(d, r)$, according to the following formula:

$$\iota(d, r) = \frac{\max\{K(r | \delta(d)), K(\delta(d) | r)\}}{\max\{K(r), K(\delta(d))\}}$$

The employment of a relative measure of inaccuracy, rather than an absolute one, facilitates the comparison of inaccuracies across different descriptions for the same representation, as well as across various descriptions for different representations.

Much like miscoding (refer to Definition 10.1.1), inaccuracy is determined using a bidirectional method: we compute the length of the shortest computer program that can generate the correct representation r given the erroneous one r' , and vice versa—namely, the computation of the shortest computer program that can generate r' given the string r . Essentially, the representation produced by a valid description needs to encompass all the necessary information for reconstructing an entity,

while it must exclude erroneous or irrelevant information.

■ **Example 11.1** Inaccuracy primarily concerns the difficulty of correcting the output of a description, that is, the output of a computable model, rather than the difficulty of rectifying the description itself. For instance, if we have a dataset generated by a system perfectly described by a quadratic function, and we choose to use a linear function as the description, inaccuracy will evaluate the original quadratic dataset against the predicted linear dataset. Inaccuracy does not measure how challenging it is to convert the incorrect linear model into the correct quadratic one. In this regard, if the original dataset comprises 10 points, a perfectly fitted polynomial of degree ten would also register an inaccuracy of zero. Deciding on the better model between the zero-inaccuracy quadratic and the zero-inaccuracy ten-degree polynomial falls under the purview of the surfeit metric (see Chapter 12). ■

Given its foundation in Kolmogorov complexity, inaccuracy is a quantity that, in general, cannot be practically computed and must be approximated. The method for approximating inaccuracy is contingent on the unique characteristics of the entities under investigation and their representations.

The inaccuracy of a description conveniently falls within the range of 0 and 1, as illustrated by the following proposition.

Proposition 11.1.1 For all representations $r \in \mathcal{B}^*$ and all descriptions $d \in \mathcal{D}$, it is true that $0 \leq \iota(d, r) \leq 1$.

Proof. This is because for all $x, y \in \mathcal{B}^*$, it follows that $0 \leq \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \leq 1$ in accordance with Proposition 6.5.3. ■

The preceding proposition applies to all possible descriptions d and representations r , even in cases where a description d is not intended to model the representation r . In such scenarios, the inaccuracy would be approximately one.

Inaccuracy is exactly zero if, and only if, the description d constitutes one of the feasible valid descriptions of the representation r .

Proposition 11.1.2 Given a description $d \in \mathcal{D}$ for a representation $r \in \mathcal{B}^*$, it is the case that $\iota(d, r) = 0$ if, and only if, d is a member of the valid descriptions of r , i.e., $d \in \mathcal{D}_r$.

Proof. If d belongs to the set \mathcal{D}_r , it follows that $K(r | \delta(d)) = K(\delta(d) | r) = 0$, implying that $\iota(d, r) = 0$. Conversely, if $\iota(d, r) = 0$, it means that $\max\{K(r | \delta(d)), K(\delta(d) | r)\} = 0$. This leads to the conclusion that $K(r | \delta(d)) = K(\delta(d) | r) = 0$ and, consequently, d is part of the set of valid descriptions for r , i.e., $d \in \mathcal{D}_r$. ■

Given two representations r and s , we want also to know the inaccuracy of the model d when describing the joint representation rs . Since we require that rs must be a valid representation, the formalization of the concept of inaccuracy applied to joint representation is straightforward, and it does not require a new definition:

$$\iota(d, rs) = \frac{\max\{K(rs | \delta(d)), K(\delta(d) | rs)\}}{\max\{K(rs), K(\delta(d))\}}$$

As a direct consequence of Proposition 11.5.1, if $r, s \in \mathcal{B}^*$ are two arbitrary representations and $d \in \mathcal{D}$ is a description, we have that $0 \leq \iota(d, rs) \leq 1$.

11.2 Conditional Inaccuracy

In this section, we delve deeper into the concept of inaccuracy by considering its application to conditional descriptions. Specifically, we explore the inaccuracy of a description when assessed in

conjunction with pre-existing background knowledge, a notion we term *conditional inaccuracy*. As we will see below, the inaccuracy of a description will never increase compare to its unconditional version; at its worst, it will simply remain constant. This property of conditional inaccuracy makes it a practical method for evaluating new concepts or models and gauging their effectiveness in explaining the entity of our interest.

In Definition 9.7.1, we introduced the concept of a conditional description d , for a representation r , given an arbitrary background string s . This is denoted by $d | s$ and was defined as the self-delimited concatenated string $\langle d, s \rangle$, where $d = \langle TM, a \rangle$ and $TM(\langle a, s \rangle) = r$. If $TM(\langle a, s \rangle) = r'$, with $r \neq r'$, then $d | s$ is referred to as an *inaccurate* conditional description of r . We also observed that $d | s$ must be defined for all possible s , and that we refer to conditioning on the empty string $d | \lambda$ as the unconditional version of d .

Drawing from the idea of inaccuracy as outlined in Definition 11.1.2, we can formulate the notion of conditional inaccuracy to encapsulate the error induced when employing an inaccurate conditional description.

Definition 11.2.1 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ a string, and $d | s$ an inaccurate conditional description. We characterize the *conditional inaccuracy* of the description d for the representation r given the string s , denoted as $\iota(d | s, r)$, according to the following equation:

$$\iota(d | s, r) = \frac{\max\{K(r | \delta(d | s)), K(\delta(d | s) | r)\}}{\max\{K(r), K(\delta(d | s))\}}$$

Conditional inaccuracy is defined as the normalized compression distance between the representation r and the string computed by the conditional description $d | s$.

As a normalized measure, the conditional inaccuracy of a description falls within the interval between 0 and 1.

Proposition 11.2.1 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ be a string, and $d | s$ be a conditional description of r given the string s . Then $0 \leq \iota(d | s) \leq 1$.

Proof. This assertion holds true given $0 \leq \frac{\max K(x|y), K(y|x)}{\max K(x), K(y)} \leq 1$ for all $x, y \in \mathcal{B}^*$, as established by Proposition 6.5.3. ■

The conditional inaccuracy assumes a null value if, and only if, the conditional description $d | s$ serves as one of the plausible valid models of the representation r .

Proposition 11.2.2 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ be a string, and $d | s$ be a conditional description of r given the string s , where $d = \langle TM, a \rangle$. Then $\iota(d | s) = 0$ if, and only if, $TM(\langle a, s \rangle) = r$.

Proof. If $TM(\langle a, s \rangle) = r$, it can be deduced that $K(r | \delta(d | s)) = K(\delta(d | s) | r) = 0$, and consequently $\iota(d | s, r) = 0$. Conversely, if $\iota(d | s, r) = 0$, we observe that $\max\{K(r | \delta(d | s)), K(\delta(d | s) | r)\} = 0$. This implies that $K(r | \delta(d | s)) = K(\delta(d | s) | r) = 0$, and therefore $TM(\langle a, s \rangle) = r$. ■

Incorporating established prior knowledge into research does not increase the inaccuracy of a description. If this background knowledge is relevant to the description of the representation, the oracle will utilize it appropriately. Conversely, if the prior knowledge is irrelevant, the oracle will simply disregard it. The following theorem formalizes this notion.

Theorem 11.2.3 Let $r \in \mathcal{B}^*$ be a representation, and $d \in \mathcal{D}$ a conditional description of r . Then

$$\iota(d | s, r) \leq \iota(d, r)$$

for all strings $s \in \mathcal{B}^*$.

Proof. Given that $\iota(d, r)$ is equivalent to $\iota(d | \lambda, r)$ we have to show

$$\frac{\max\{K(r | \delta(d | s)), K(\delta(d | s) | r)\}}{\max\{K(r), K(\delta(d | s))\}} \leq \frac{\max\{K(r | \delta(d | \lambda)), K(\delta(d | \lambda) | r)\}}{\max\{K(r), K(\delta(d | \lambda))\}}$$

This inequality follows from the fact that $K(r | \langle \delta(d), s \rangle) \leq K(r | \delta(d))$, as demonstrated in Proposition ??.

Theorem 11.2.3 represents a seminal result in the theory of nescience. It lays the groundwork for the development of a robust methodology to deepen our understanding (i.e., reduce inaccuracy) of a research entity. In practical contexts, our main focus will be on prior knowledge that directly pertains to our study. However, the essence of Theorem 11.2.3 is its indication that we can venture into concepts from seemingly unrelated domains without affecting our primary investigation. This theorem becomes especially valuable when such explorations are automated (see Chapter 20).

■ **Example 11.2** The P vs NP problem stands as a pivotal unresolved question in computer science. It seeks to determine whether every problem, whose solution can be verified quickly (in polynomial time), can also be solved in a similar expedited manner. The intricate relationship between these two classes, P (problems solvable quickly) and NP (problems verifiable quickly), remains undeciphered. Providing a comprehensive, self-contained solution to this problem in formal language could be a daunting task. However, drawing upon pre-existing knowledge can significantly condense the solution. For instance, incorporating insights from Algorithm Theory, which elucidates the classification and efficiency of algorithms, or leveraging principles from Formal Language Theory, which maps out the categorization and approach to different computational challenges such as regular or context-free languages, and underscores the significance of Turing machines, can be immensely beneficial. Moreover, harnessing and building upon established background knowledge might not only simplify our descriptions but could also pave the way for a deeper understanding and potential resolution of the P vs. NP conundrum. ■

Finally, given two representations r and t , the formalization of the concept of conditional inaccuracy, when applied to the joint representation rt , is quite straightforward, and it does not demand a new definition:

$$\iota(d | s) = \frac{\max\{K(rt | \delta(d | s)), K(\delta(d | s) | rt)\}}{\max\{K(rt), K(\delta(d | s))\}}$$

11.3 Decreasing Inaccuracy

Our objective is to reduce the inaccuracy of our current description d_1 , thereby enhancing our understanding of the original entity. This enhancement might take the form of a modified version of d_1 , correcting or eliminating its errors, or by formulating an entirely new description using a different approach to model the entity. In either scenario, the revised or new version is considered as a new description d_2 . In this section, we aim to analyze how introducing a new description d_2 affects the inaccuracy relative to the original description d_1 .

Definition 11.3.1 Let $r \in \mathcal{B}^*$ be a representation, and $d_1, d_2 \in \mathcal{D}$ be two descriptions. The variation of the inaccuracy of the descriptions d_1, d_2 , denoted by $\Delta_t^a(d_1, d_2, r)$, is defined as:

$$\Delta_t^a(d_1, d_2, r) = \iota(d_1, r) - \iota(d_2, r)$$

Since inaccuracy can't exceed 1 or go below 0, the maximum possible variation of the inaccuracy is 1. A positive value of Δ_t suggests a preference for description d_2 over d_1 . Conversely, a negative

value indicates the reverse. This preference is strictly based on inaccuracy. It's possible that the new description could result in a significant increase in surfeit, which might be even more significant than the reduction in inaccuracy (refer to Chapter 12 for a deeper discussion on surfeit, and Chapter 14 to understand how these two concepts merge into a single metric termed nescience).

We can also introduce a relative variation of inaccuracy.

Definition 11.3.2 Let $r \in \mathcal{B}^*$ be a representation, and $d_1, d_2 \in \mathcal{D}$ be two descriptions. The *relative variation of the inaccuracy* of the descriptions d_1, d_2 , denoted by $\Delta_i^r(d_1, d_2)$, is defined as:

$$\Delta_i^r(d_1, d_2, r) = \frac{i(d_1, r) - i(d_2, r)}{i(d_1, r)}$$

We are primarily concerned with descriptions d_2 that offer improvements over the existing description d_1 . This means that the inaccuracy of d_2 is less than that of d_1 . Accordingly, the relative variation ranges from 0, indicating no improvement in inaccuracy at all, to 1, denoting maximal improvement. Regrettably, Δ_i^r can also be negative when the new description d_2 is even worse than the original d_1 .

As the inaccuracy gets closer to zero, relative variations become more volatile. A small absolute variation can result in a very large relative variation if the initial inaccuracy is tiny. For example, if $i(d_1, r) = 0.1$ and the inaccuracy reduces by an absolute amount of 0.05, this corresponds to a relative variation of 50%, but if $i(d_1, r) = 0.1$ it is only a 5.6% relative reduction. Both absolute and relative variations remain crucial for understanding the magnitude and significance of changes.

An alternative method to reduce uncertainty about an entity might involve modifying its representation rather than adjusting its description. While this could lead to an increase in the entity's miscoding, the potential decrease in inaccuracy could compensate for this drawback.

Definition 11.3.3 Let $r_1, r_2 \in \mathcal{B}^*$ be two representations, and $d \in \mathcal{D}$ be a description. The *variation of the inaccuracy* of the representations r_1, r_2 , denoted by $\Delta_i^a(d, r_1, r_2)$, is defined as:

$$\Delta_i^a(d, r_1, r_2) = i(d, r_1) - i(d, r_2)$$

Again, and since inaccuracy can't exceed 1 or go below 0, the maximum possible variation of the inaccuracy is 1. A positive value of Δ_i^a suggests a preference for representation r_2 over r_1 . Conversely, a negative value indicates the reverse. This preference is strictly based on inaccuracy, since miscoding is not taken into account. There is no risk of a change in surfeit, since the description does not change.

We can also introduce a relative variation of inaccuracy.

Definition 11.3.4 Let $r_1, r_2 \in \mathcal{B}^*$ be two representations, and $d \in \mathcal{D}$ be a description. The *relative variation of the inaccuracy* of the representations r_1, r_2 , denoted by $\Delta_i^r(d, r_1, r_2)$, is defined as:

$$\Delta_i^r(d, r_1, r_2) = \frac{i(d, r_1) - i(d, r_2)}{i(d, r_1)}$$

We are primarily concerned with a representation r_2 that offer improvements over the existing representation r_1 . This means that the inaccuracy of using r_2 is less than that of r_1 . Accordingly, the relative variation ranges from 0, indicating no improvement in inaccuracy at all, to 1, denoting maximal improvement. Regrettably, Δ_i^r can also be negative when the new representation r_2 is even worse than the original r_1 . As the inaccuracy gets closer to zero, relative variations become more volatile. A small absolute variation can result in a very large relative variation if the initial inaccuracy is tiny.

11.4 Inaccuracy-Miscoding Rate of Change

In the preceding section, we discussed how, given a specific representation, one can reduce its inaccuracy by adopting a different description. We also delved into an alternative approach, where the inaccuracy is minimized not by altering the description, but by employing a modified representation. In this section, we turn our attention to a more generic strategy for decreasing inaccuracy tied to an entity. Rather than individually modifying the description or representation, there might be greater benefits in adjusting both simultaneously. The compromise between the amount of miscoding we are willing to accept in order to improve inaccuracy is termed the miscoding-inaccuracy trade-off (for a deeper understanding of trade-offs in multi-objective optimization, please refer to Section 7.5.2).

Definition 11.4.1 Let $e \in \mathcal{E}$ be an entity, and $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{R}_e \times \mathcal{D}$ be two hypothesis, with $\mathbf{x}_1 = (r_1, d_1)$ and $\mathbf{x}_2 = (r_2, d_2)$. The *rate of change between the inaccuracy and the miscoding* of the hypothesis $\mathbf{x}_1, \mathbf{x}_2$, denoted by $\Delta_{\mu\iota}(\mathbf{x}_1, \mathbf{x}_2)$, is defined as:

$$\Delta_{\mu\iota}(\mathbf{x}_1, \mathbf{x}_2) = \frac{\iota(d_1, r_1) - \iota(d_2, r_2)}{\mu(r_2) - \mu(r_1)}$$

assuming that $\mu(r_2) - \mu(r_1) \neq 0$.

The ratio, $\Delta_{\mu\iota}$, represents the rate of change between inaccuracy and miscoding when transitioning from the first hypothesis to the second. A positive value of $\Delta_{\mu\iota}$ implies that either both quantities—miscoding and inaccuracy—decrease (which is beneficial), or both increase (which is undesirable). The complexity arises when $\Delta_{\mu\iota}$ is negative, indicating that one of the quantities decreases while the other increases. In such situations, we have two possible scenarios:

- (i) If inaccuracy decreases and miscoding increases, we aim for $\Delta_{\mu\iota}(\mathbf{x}_1, \mathbf{x}_2) < M$, where $M < -1$, ensuring that the decrease in inaccuracy compensates for the rise in miscoding.
- (ii) If inaccuracy increases while miscoding decreases, we target $\Delta_{\mu\iota}(\mathbf{x}_1, \mathbf{x}_2) > M$, where $-1 < M < 0$, ensuring the reduction in miscoding offsets the increase in inaccuracy.

In both cases, it's essential to be cautious when the change in miscoding is minimal, as it can distort the results.

■ **Example 11.3** Let \mathbf{x}_1 and \mathbf{x}_2 be two hypotheses. For \mathbf{x}_1 we have that the inaccuracy $\iota(d_1, r_1)$ is 0.40, and the miscoding $\mu(r_1)$ is 0.15. In case of \mathbf{x}_2 , the inaccuracy $\iota(d_2, r_2)$ is 0.20 (a decrease from 0.40), and miscoding $\mu(r_2)$ is 0.25 (an increase from 0.15). Using the definition of rate of change we have that:

$$\Delta_{\mu\iota}(\mathbf{x}_1, \mathbf{x}_2) = \frac{0.40 - 0.20}{0.15 - 0.25} = \frac{0.20}{-0.10} = -2$$

In this case, when transitioning from hypothesis \mathbf{x}_1 to \mathbf{x}_2 , the inaccuracy decreased by 0.20 units, while miscoding increased by 0.10 units. The rate of change is -2. ■

Having a very small change in miscoding can significantly amplify the value of the rate of change, which might give an impression that inaccuracy and miscoding are changing at an extreme rate, even if they are not. Let's illustrate this with an example.

■ **Example 11.4** Let \mathbf{x}_1 be an hypothesis with an inaccuracy $\iota(d_1, r_1)$ of 0.35, and a miscoding $\mu(r_1)$ of 0.20. And let \mathbf{x}_2 be a second hypothesis with an inaccuracy $\iota(d_2, r_2)$ of 0.30 (a small decrease from 0.35), and a miscoding $\mu(r_2)$ of 0.2001 (a very tiny increase from 0.20). Applying the definition of rate of change:

$$\Delta_{\mu\iota}(\mathbf{x}_1, \mathbf{x}_2) = \frac{0.35 - 0.30}{0.20 - 0.2001} = \frac{0.05}{-0.0001} = -500$$

The rate of change is -500, which might give the impression that the change in inaccuracy and miscoding was very large. However, in reality, inaccuracy only decreased by 0.05 units and miscoding increased by a negligible 0.0001 units. The extremely small denominator exaggerated the rate of change, making it appear misleadingly large. ■

As you try to optimize both inaccuracy and miscoding, there will be certain configurations (hypotheses) where you can't improve inaccuracy without increasing miscoding, or vice versa. The collection of these "best trade-off" configurations forms the Pareto frontier (see Section XX). Points on the Pareto frontier are Pareto optimal because any small change to move away from one of these points would result in a degradation of one of the objectives without an improvement in the other.

Definition 11.4.2 Let $e \in \mathcal{E}$ be an entity, and let $\mathbf{x} = (r, d) \in \mathcal{R}_e \times \mathcal{D}$ be a hypothesis. The hypothesis \mathbf{x} is said to be a Pareto point with respect to inaccuracy ι and miscoding μ if there does not exist another hypothesis $\mathbf{x}' = (r', d')$ such that:

- 1 $\iota(d', r') \leq \iota(d, r)$ and $\mu(r') \leq \mu(r)$, and
- 2 $\iota(d', r') < \iota(d, r)$ or $\mu(r') < \mu(r)$.

In simpler terms, a hypothesis \mathbf{x} is Pareto optimal if: i) no other hypothesis is better in terms of both inaccuracy and miscoding; ii) at least for one of the two (either inaccuracy or miscoding), no other hypothesis is strictly better.

The rate of change, $\Delta_{\iota\mu}$, between any two Pareto optimal points would give insights into how the trade-offs between inaccuracy and miscoding vary along the Pareto front. If the decision-makers are more sensitive to changes in inaccuracy than miscoding, they might opt for configurations with a less negative $\Delta_{\iota\mu}$ value. Conversely, if miscoding is of greater concern, they might accept configurations where $\Delta_{\iota\mu}$ suggests a larger rise in inaccuracy for a smaller improvement in miscoding.

11.5 Inaccuracy of Areas

An area \mathcal{A} is a subset $\mathcal{A} \subset \mathcal{E}$ of entities that are related or share a common property. The concept of inaccuracy can be extended to research areas in order to quantitatively measure the amount of effort required to fix an inaccurate description of an area.

Given the strings r_1, r_2, \dots, r_n , where each r_i belongs to \mathcal{B}^* for $i = 1, 2, \dots, n$, recall that we use $\langle r_1, r_2, \dots, r_n \rangle$ to represent a re-encoded version of the individual strings r_i into one unified string, in such a way that it can be decomposed back into the original strings.

The following definition extends the concept of inaccuracy to areas.

Definition 11.5.1 Let $\mathcal{A} \subset \mathcal{E}$ be an area with known subset $\hat{\mathcal{A}} = \{r_1, r_2, \dots, r_n\}$, and $d \in \mathcal{D}$ a description. We define the *inaccuracy of the area* given the description d , denoted by $\iota(d, \hat{\mathcal{A}})$, as:

$$\iota(d, \hat{\mathcal{A}}) = \frac{\max\{K(\langle r_1, r_2, \dots, r_n \rangle | \delta(d)), K(\delta(d) | \langle r_1, r_2, \dots, r_n \rangle)\}}{\max\{K(\langle r_1, r_2, \dots, r_n \rangle), K(\delta(d))\}}$$

The inaccuracy of the description of an area falls within the range of 0 and 1, as illustrated by the following proposition.

Proposition 11.5.1 For all known subsets $\hat{\mathcal{A}} = \{r_1, r_2, \dots, r_n\}$, and all $d \in \mathcal{D}$ descriptions, we have that $0 \leq \iota(d, \hat{\mathcal{A}}) \leq 1$.

Proof. Given that $\langle r_1, r_2, \dots, r_n \rangle$ is a string and Proposition 6.5.3. ■

By extending the concept of inaccuracy to encompass areas, we can quantitatively evaluate the quality of descriptions for specific subsets of entities. This mathematical framework offers a robust tool to assess and rectify inaccuracies in both individual entities and broader research areas.

References

A good introduction to the study of uncertainties (error analysis in models) in science, and in particular in physics, chemistry, and engineering, is the best-selling text [[taylor2022introduction](#)], which also features the same image of a crashed train than in the introduction to this chapter. Another excellent introduction to error analysis intended for undergraduate students in science or technology is [[hughes2010measurements](#)]. From a more philosophical point of view, we have the work of Popper [[popper2014conjectures](#)] in which he introduces the concept of falsifiability, arguing that for a theory to be considered scientific, it must be testable and refutable.



12. Surfeit

*Everything should be made as simple as possible,
but not simpler.*
Albert Einstein

Surfeit is the final metric we will introduce to quantitatively measure our understanding of a research entity. It quantifies the presence of superfluous symbols in the description we use to model such an entity. Intuitively, our lack of knowledge about the entity is typically reflected in the length (number of symbols) of our prevailing description. Lengthy descriptions often include erroneous or redundant elements. As we enhance our understanding of the subject, we should be able to identify and eliminate these unnecessary symbols, leading to a more concise and precise description.

We define the surfeit of a description for an entity as the discrepancy in length between the given description and the optimal one for that entity. Within the framework of the theory of nescience, we operate under the assumption that the epitome of knowledge about an entity, or its perfect description, is encapsulated in the briefest description that enables the complete reconstruction of an entity's representation. This notion of perfection is contingent upon the validity of the representation and the accuracy of the description.

The length of the most concise description of an entity is determined by the Kolmogorov complexity of a representation for that entity. As previously discussed, Kolmogorov complexity is not computable in general. Additionally, in practical terms, and considering that our knowledge about entities is typically incomplete, the most concise possible description remains unknown. As a result, surfeit is a metric that requires approximation in practice.

If we could devise a perfect description of an entity, such a description would have to be a random string; otherwise, it would embody redundant elements that could be omitted. In the context of the theory of nescience, attaining perfect knowledge is equivalent to achieving a state of randomness. This intrinsic randomness delineates a boundary on the depth of understanding attainable for a specific research topic. However, far from being a limitation, the acknowledgment and comprehension of this boundary unveil unprecedented opportunities in both science and technology. For instance, by assessing the deviation of our current description from a random string,

we can estimate the closeness to a perfect description's realization.

In this chapter, we will formally introduce the concept of surfeit and delve into its properties, including conditional surfeit. The concept of redundancy will also be unveiled as a practical approximation to surfeit. We will explore strategies to mitigate both surfeit and redundancy and scrutinize the relation between a reduction in surfeit and variations in accuracy or miscoding. Lastly, we will extend the application of the surfeit concept to incorporate the analysis of entire research areas.

12.1 Surfeit

Given the length of a description of a representation for an entity and the length of its shortest possible description, we can introduce a relative measure to quantify the unnecessary effort expended in explaining the entity with that particular description. We term this quantity *surfeit*, and it constitutes an integral component of our definition of nescience, representing the extent of our ignorance about the research entity.

Definition 12.1.1 — Surfeit. Given a representation $r \in \mathcal{B}^*$, and $d \in \mathcal{D}$ a description for r , we define the *surfeit of the description d for the representation r* , denoted by $\sigma(d, r)$, as

$$\sigma(d, r) = \frac{|l(d) - K(r)|}{l(d)}$$

For the majority of descriptions, it will be the case that the length of the description $l(d)$ for r exceeds the length of its shortest possible description $K(r)$. Intuitively, the more ignorant we are about an entity, the longer our description will be. A refined understanding of the entity should enable us to eliminate all redundant elements from its description. However, there might also be instances where the description is shorter than the optimal one. In such cases, we are oversimplifying the problem, which is equally detrimental. This is the rationale behind using the absolute value $|l(d) - K(r)|$ instead of simply $l(d) - K(r)$. Naturally, the current description might also be inaccurate, and the representation could be invalid. However, these issues are addressed by the metrics of inaccuracy and miscoding.

In our definition of surfeit, we opted for a relative measure rather than an absolute one (i.e., $|l(d) - K(r)|$) because we are interested not only in comparing the surfeit of different models for the same entity but also in comparing the surfeit across different entities. We prefer to use $K(r)$ instead of the equivalent $l(r^*)$ to maintain consistency with the definition of inaccuracy provided in Section 11.1.

■ **Example 12.1** In a practical machine learning scenario, consider a task of classifying images of cats and dogs, being our representation a set of training images. An initial complex model, encumbered with excessive parameters and unnecessary features, represents a lengthy "description" of the problem. The optimal model, however, balances accuracy and simplicity. The surfeit measures the excess complexity of the initial model compared to this optimal one. It quantifies the "extra" elements that aren't essential for accurate classification. A high surfeit indicates an overly complicated model, signaling the need for refinement to achieve efficiency and effectiveness, a principle crucial in machine learning for developing models that generalize well to unseen data. ■

The surfeit of a description is a number between 0 and 1.

Proposition 12.1.1 Let $r \in \mathcal{B}^*$ be a representation, and $d \in \mathcal{D}_r^*$ one of its **XXX valid descriptions**, then we have that $0 \leq \sigma(d, r) \leq 1$.

Proof. Given that $l(d) > 0$ and $K(r) > 0$, as they are the lengths of non-empty strings, and $l(d) \geq K(r)$ because we are not considering **XXX pleonastic descriptions**. ■

The surfeit is zero when the length of the description $l(d)$ is equal to the Kolmogorov complexity $K(r)$ of the representation of the entity, signifying that the description has achieved theoretical conciseness.

Proposition 12.1.2 Let $r \in \mathcal{B}^*$ be a representation, and $d \in \mathcal{D}$ a description for r , then we have that $\sigma(d, r) = 0$ if and only if $l(d) = l(d^*)$.

Proof. As a direct consequence of the definition of Kolmogorov complexity. ■

It is essential to distinguish between conciseness and correctness. A zero surfeit highlights the optimal brevity of the description, but not necessarily its accuracy. The accuracy, or correctness, of the description is evaluated by the metric inaccuracy. Thus, while a zero surfeit points to a minimally lengthy, streamlined description, the inaccuracy metric is consulted to determine the extent to which this concise description is correct and reliable. Together, surfeit and inaccuracy offer a comprehensive assessment of the description's efficiency and validity.

■ **Example 12.2** In a machine learning scenario, a model designed to classify emails as spam or not has achieved a surfeit of zero, indicating optimal conciseness with no superfluous elements. However, despite its streamlined complexity, the model is found to have high inaccuracy, frequently misclassifying emails. This illustrates the dichotomy between surfeit and inaccuracy. While the model is theoretically as concise as possible, signified by the zero surfeit, its practical application is hampered by incorrect classifications, highlighted by the inaccuracy metric. This underscores the necessity to balance a minimal surfeit with low inaccuracy to achieve a model that is both efficient and accurate. ■

Explain that there could be more than one description that makes surfeit zero for the same representation.

Our definition of surfeit compares the length of a description with the Kolmogorov complexity of the representation, not with the Kolmogorov complexity of the description itself (i.e., $K(d)$). That is, surfeit is not a measure of the redundancy of a description. It might happen that we come up with an incompressible description (no redundant elements to remove), that it is not the shortest possible one that describes the representation (see Example 9.13). Such a description would not be redundant in the traditional sense, but it still will present some surfeit in the sense of the theory of nescience. Moreover, it might happen that the description d we are considering does not describe the representation r , that is, $d \in \mathcal{D}_r$. In practice it is highly convenient to introduce the following alternative (we could say weaker) characterization of the concept of redundant model:

Definition 12.1.2 — Redundancy. Given a description $d \in \mathcal{D}$, we define the *redundancy* of the description d , denoted by $\rho(d)$, as

$$\rho(d) = 1 - \frac{K(d)}{l(d)}$$

The redundancy of a description d is a quantity related to the description itself, and it does not depend on the representation r being described.

■ **Example 12.3 TODO: Provide an example to clarify the concept.** ■

We have that the redundancy of a description is a number between 0 and 1.

Proposition 12.1.3 We have that $0 \leq \rho(d) \leq 1$ for all $d \in \mathcal{D}$.

Proof. Apply Proposition 6.2.2. ■

Clarify when the redundancy is zero

Finally, next proposition formalizes our intuition that the surfeit of a description is greater or equal than its redundancy.

Proposition 12.1.4 Let $r \in \mathcal{B}^*$ be a representation, and $d \in \mathcal{D}_r^*$ one of its valid descriptions, then we have that $\rho(d) \leq \sigma(d, r)$.

Proof. Proving that $\rho(d) \leq \sigma(d, r)$ is equivalent to prove that $K(d) \geq K(r)$ for all d . Lets assume that there exist a d such that $K(d) < K(r)$, that would mean there exists a Turing machine $\langle TM, a \rangle$ such that $TM(a) = r$ but $l(\langle TM, a \rangle) < K(r)$. That is a contradiction with the fact that $K(r)$ is the length of the shortest possible Turing machine that prints r . ■

It would be very nice if Proposition 12.1.4 applies to all possible description. Unfortunately, the proposition is true only when we deal with valid descriptions (from \mathcal{D}_r^*), as Example 12.4 shows.

■ **Example 12.4** TODO: Provide an example. ■

12.2 Conditional Surfeit

We are interested into study how the surfeit of a description for a representation is affected when some background knowledge is assumed. That is, we want to know the surfeit of a conditional description for a representation, what we call *conditional surfeit*.

Definition 12.2.1 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ be a string, and $d \in \mathcal{D}$ be a description of r given s . We define the *conditional surfeit* of the conditional description $d_{r|s}$, denoted by $\sigma(d_{r|s})$, as:

$$\sigma(d_{r|s}) = 1 - \frac{K(r|s)}{l(d_{r|s})}$$

This definition is required mostly for practical purposes, since given that our knowledge of s is perfect, we can focus on studying what it is new in topic t , that is, in that part not covered by the assumed (perfect) background knowledge.

Conditional surfeit, being a relative measure, is a number between 0 and 1.

Proposition 12.2.1 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ be a string, and $d \in \mathcal{D}$ be a description of r given s . We have that $0 \leq \sigma(d_{r|s}) \leq 1$.

Proof. Given that $l(d_{r|s}) > 0$ and that $K(r|s) > 0$, since they are the lengths of non-empty strings, and that $l(d) \geq K(r)$ since we do not consider pleonastic descriptions. ■

Intuition tell us that the surfeit of a description could only decrease if we assume the background knowledge given by the description of another topic. This is because we require that this background knowledge must be a perfect description (it presents no surfeit). However, as it was the case of joint surfeit, we have to wait until Chapter 14 to formalize this intuition.

In the same way we introduced the concept of redundancy of a description as a weaker version of the concept of surfeit, we can also introduce the concept of conditional redundancy as a weaker version of the concept of conditional surfeit.

Definition 12.2.2 Let $s \in \mathcal{B}^*$ be a string, and $d \in \mathcal{D}$ be a conditional description given s . We define the *conditional surfeit* of the conditional description $d_{t|s^*}$, denoted by $\sigma(d_{t|s^*})$, as:

$$\rho(d_{r|s}) = 1 - \frac{K(d_{t|s^*})}{l(d_{t|s^*})}$$

Conditional surfeit is a relative measure, and so, a number between 0 and 1.

Proposition 12.2.2 We have that $0 \leq \rho(d_{t|s^*}) \leq 1$ for all t, s and all $d_{t,s}$.

Proof. Given that $K(d_{t|s^*}) \leq l(d_{t|s^*})$ we have that $\frac{K(d_{t|s^*})}{l(d_{t|s^*})} \leq 1$ and so, $1 - \frac{K(d_{t|s^*})}{l(d_{t|s^*})} \geq 0$. Also, since $\frac{K(d_{t|s^*})}{l(d_{t|s^*})} > 0$ (both quantities are positive integers), we have that $1 - \frac{K(d_{t|s^*})}{l(d_{t|s^*})} \leq 1$. ■

Finally, we can extend our concepts of conditional surfeit and conditional redundancy to multiple, but fine, number of topics.

Definition 12.2.3 Let $t, s_1, s_2, \dots, s_n \in \mathcal{T}$ be a finite collection of topics, and let $d_{t|s_1^*, s_2^*, \dots, s_n^*}$ any conditional description of t given s_1, s_2, \dots, s_n . We define the *conditional surfeit* of the description $d_{t|s_1^*, s_2^*, \dots, s_n^*}$, denoted by $\sigma(d_{t|s_1^*, s_2^*, \dots, s_n^*})$, as:

$$\sigma(d_{t|s_1^*, s_2^*, \dots, s_n^*}) = 1 - \frac{K(t | s_1^*, s_2^*, \dots, s_n^*)}{l(d_{t|s_1^*, s_2^*, \dots, s_n^*})}$$

And the *conditional redundancy* of the description d_{t_1, t_2, \dots, t_n} , denoted by $\rho(d_{t_1, t_2, \dots, t_n})$, as:

$$\rho(d_{t_1, t_2, \dots, t_n}) = 1 - \frac{K(d_{t_1, t_2, \dots, t_n})}{l(d_{t_1, t_2, \dots, t_n})}$$

It is easy to show that the properties of conditional surfeit and conditional redundancy apply to the case of multiple topics as well.

12.3 Decreasing Surfeit

[Write this section](#)

12.4 Rate of Change

[Write this section](#)

12.5 Surfeit of Areas

[Review this section](#)

The concept of surfeit can be extended to research areas, to quantitative measure the amount of extra effort we are using to describe the topics of the area.

Definition 12.5.1 Let $A \subset \mathcal{T}$ be an area with known subset $\hat{A} = \{t_1, t_2, \dots, t_n\}$, and let $d_{\hat{A}}$ be a description. We define the *surfeit of the description* $d_{\hat{A}}$ as:

$$\sigma(d_{\hat{A}}) = 1 - \frac{K(\langle t_1, t_2, \dots, t_n \rangle)}{l(d_{\hat{A}})}$$

As it was the case of the concept of redundancy, in general we do not know the complexity of the area $K(\hat{A})$, and so, in practice, it must be approximated by the complexity of the descriptions themselves $K(\hat{d}_{\hat{A}})$. However, in the particular case of areas, we could have also problems with the quantity $\hat{d}_{\hat{A}}$, since it requires to study the conditional descriptions of the topics included in the area.

Definition 12.5.2 Let $A \subset \mathcal{T}$ be an area with known subset $\hat{A} = \{t_1, t_2, \dots, t_n\}$, and let $d_{\hat{A}}$ be a

description. We define the *weak redundancy of the description* $d_{\hat{A}}$ as:

$$\rho(d_{\hat{A}}) = 1 - \frac{K(d_{\hat{A}})}{l(d_{\hat{A}})}$$

References

The concept of redundancy has been also investigated in the context of information theory, since we are interested on using codes with low redundancy (see for example [[abramson1963information](#)]).

13. Mismodel

*Everything should be made as simple as possible,
but not simpler.*
Albert Einstein

TODO: Nice introduction

Mismodel, being a non-computable quantity, has to be approximated in practice. We propose to approximate the concept of mismodel by splitting it into two complementary subconcepts: inaccuracy and surfeit. Inaccuracy compares the output of our description with the representation selected to encode the entity.

13.1 Mismodel

In Section 9.5 we defined the concept of description, or model, of an entity as a computer program that, when executed, recreates one of the representations that encode that entity. More specifically, a description d of a representation r encoding an entity e is a Turing machine that, when interpreted by a universal Turing machine δ , prints out the string r . We also saw that the objective of science is to find from the shortest description d_r^* from the collection of perfect descriptions \mathcal{D}_r^* that model the representation r . This shortest description d_r^* might not be unique.

Since our knowledge about the entity e under study is incomplete, and in particular our knowledge about the representation r of e is incomplete, we will be usually working with a description d that is, hopefully, close to one of the shortest descriptions d_r^* , but not necessarily equal. We are interested in to quantify the error due to the use of a description d that is not perfect.

We define the mismodel of a description for a representation as the normalized information distance between our description and all possible minimal descriptions of r .

Definition 13.1.1 — Mismodel. Let $r \in \mathcal{B}^*$ be a representation, and $d \in \mathcal{D}$ be a description. We

define the *mismodel* of the description d for the representation r , denoted by $\sigma(d, r)$, as:

$$\sigma(d, r) = \min_{d^* \in d_r^*} \frac{\max\{K(d | d^*), K(d^* | d)\}}{\max\{K(d), K(d^*)\}}$$

The description d we are using not only it might not be one of the shortest possible models that print out r , but in fact, it might happen that what d prints is not r .

We cannot use the more ...

The mismodel of a description is, conveniently, a number between 0 and 1, as next proposition shows.

Proposition 13.1.1 We have that $0 \leq \sigma(d, r) \leq 1$ for all representations $r \in \mathcal{B}^*$ and all the descriptions $d \in \mathcal{D}$.

Proof. Given that $0 \leq \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \leq 1$ for all $x, y \in \mathcal{B}^*$ according to Proposition 6.5.3. ■

The above proposition holds for all possible descriptions d and all possible representations r , even in the case that a description d is not intended as a model for the representation r , in which case the inaccuracy would be close to one.

Mismodel is equal to zero if, and only if, the description d is one of the possible valid descriptions of the representation r .

Proposition 13.1.2 Let $d \in \mathcal{D}$ be a description for a representation $r \in \mathcal{B}^*$, we have that $\iota(d, r) = 0$ if, and only if, $d \in \mathcal{D}_r$.

Proof. If $d \in \mathcal{D}_r$ we have that $K(r | \delta(d)) = K(\delta(d) | r) = 0$ and that $\iota(d, r) = 0$. If $\iota(d, r) = 0$ we have that $\max\{K(r | \delta(d)), K(\delta(d) | r)\} = 0$, which implies that $K(r | \delta(d)) = K(\delta(d) | r) = 0$ and that $d \in \mathcal{D}_r$. ■

13.1.1 Mismodel of Joint Representations

Given two representations r and s , we want to know the inaccuracy of the model d when describing the joint representation rs . Since we require that rs must be a valid representation, the formalization of the concept of inaccuracy applied to joint representation is straightforward, and it does not require a new definition:

$$\iota(d, rs) = \frac{\max\{K(rs | \delta(d)), K(\delta(d) | rs)\}}{\max\{K(rs), K(\Gamma(d))\}}$$

As a direct consequence of Proposition 10.1.1, if $r, s \in \mathcal{B}^*$ are two arbitrary representations and $d \in \mathcal{D}$ is a description, we have that $0 \leq \iota(d, rs) \leq 1$.

TODO: Recall the properties of NID, and particularize for the case of inaccuracy of joint representations.

13.1.2 Conditional Mismodel

In this section we are going to extend the concept of inaccuracy from descriptions to conditional descriptions, that is, the inaccuracy of a description assuming the existence of some background knowledge, what we call conditional inaccuracy.

We have to start by defining what we mean when we say that a conditional description is inaccurate.

Definition 13.1.2 Let $r \in \mathcal{B}^*$ be a representation, and $d_{r|s} = \langle d, s \rangle$ a conditional description of r given the string $s \in \mathcal{B}^*$, with $d = \langle TM, a \rangle$. If $TM(\langle a, s \rangle) = r'$, such that $r \neq r'$, we say that $d_{r|s}$ is an *inaccurate conditional description* for r .

In the same way we defined the concept of inaccuracy of a description in Definition 11.1.2, we can define the concept of conditional inaccuracy to characterize the error made when using an inaccurate conditional description.

Definition 13.1.3 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ a string, and $d_{r|s} = \langle d, s \rangle$ a inaccurate conditional description. We define the *conditional inaccuracy* of the description $d_{r|s}$ for the representation r given the string s , denoted by $\iota(d_{r|s})$, as:

$$\iota(d_{r|s}) = \frac{\max\{K(r | \delta(\langle d, s \rangle)), K(\delta(\langle d, s \rangle) | r)\}}{\max\{K(r), K(\delta(\langle d, s \rangle))\}}$$

The conditional inaccuracy of a description is a number between 0 and 1.

Proposition 13.1.3 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ a string, and $d_{r|s} = \langle d, s \rangle$ a inaccurate conditional description. We have that $0 \leq \iota(d_{r|s}) \leq 1$.

Proof. Given that $0 \leq \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \leq 1$ for all $x, y \in \mathcal{B}^*$ according to Proposition 6.5.3. ■

Conditional inaccuracy is equal to zero if, and only if, the description d is one of the possible valid descriptions of the representation r .

Proposition 13.1.4 Let $r \in \mathcal{B}^*$ be a representation, $s \in \mathcal{B}^*$ a string, and $d_{r|s} = \langle d, s \rangle$ a conditional description, with $d = \langle TM, a \rangle$. We have that $\iota(d_{r|s}) = 0$ if, and only if, $TM(\langle a, s \rangle) = r$.

Proof. If $TM(\langle a, s \rangle) = r$ we have that $K(r | \delta(d_{r|s})) = K(\delta(d_{r|s}) | r) = 0$ and that $\iota(d_{r|s}) = 0$. If $\iota(d_{r|s}) = 0$ we have that $\max\{K(r | \delta(d_{r|s})), K(\delta(d_{r|s}) | r)\} = 0$, which implies that $K(r | \delta(d_{r|s})) = K(\delta(d_{r|s}) | r) = 0$ and that $TM(\langle a, s \rangle) = r$. ■

Given two representations r and s , we want to know the inaccuracy of a conditional model d geniven t when describing the joint representation rs . Since we require that rs must be a valid representation, the formalization of the concept of contional inaccuracy applied to joint representation is straightforward, and it does not require a new definition:

$$\iota(d_{rs|t}) = \frac{\max\{K(r | \delta(\langle d, t \rangle)), K(\delta(\langle d, t \rangle) | r)\}}{\max\{K(rs), K(\delta(\langle d, t \rangle))\}}$$

As a direct consequence of Proposition 13.1.3, if $r, s \in \mathcal{B}^*$ are two arbitrary representations, $d_{r|s} = \langle d, s \rangle$ is a conditional description and $t \in \mathcal{B}^*$ an arbitrary string, we have that $0 \leq \iota(d_{rs|t}) \leq 1$.

13.2 Reducing Mismodel

A valid representation of an entity is a string that contains all the necessary information required by the oracle to reconstruct that entity and only that information. If the representation is non-valid, meaning its miscoding exceeds zero, it may be because some crucial information is missing, some symbols are incorrect, or it contains irrelevant symbols. To reduce the miscoding of a representation, we can add the missing information, or remove the incorrect or irrelevant symbols. We cannot know in advance if any information is missing or some symbols need to be removed. Fortunately, as the next theorem illustrates, it must be one of these two cases.

Theorem 13.2.1 Let $r \in \mathcal{B}^*$ be a representation such that $\mu(r) > 0$, then at least one of the following cases is true:

- (i) there exist a $s \in \mathcal{B}^*$ such that $\mu(rs) < \mu(r)$ or $\mu(sr) < \mu(r)$,
- (ii) there exists a $s \in \mathcal{B}^*$ in the form $r = \alpha s \beta$ with $\alpha, \beta \in \mathcal{B}^*$ such that $\mu(r) < \mu(s)$.

Proof. Finish Assume that $\mu(r) > 0$. We have that

$$\min_{r_e^* \in \mathcal{R}_e^*} \frac{\max\{K(r | r_e^*), K(r_e^* | r)\}}{\max\{K(r), K(r_e^*)\}} > 0$$

Let $r_e^* = \arg \min(\mu(r))$. We have that $\max\{K(r | r_e^*), K(r_e^* | r)\} > 0$. If $K(r | r_e^*) > 0$ we have that r contains non-relevant symbols. If $K(r_e^* | r) > 0$ we have that r is missing some relevant symbols. ■

■ **Example 13.1** TODO: Provide a practical example. ■

13.3 Mismodel of Areas

The concept of miscoding can be extended to research areas in order to quantitative measure the amount of effort required to fix an inaccurate representation of the area.

■ **Definition 13.3.1** Let $A \subset \mathcal{E}$ be an area with known subset $\hat{A} = \{e_1, e_2, \dots, e_n\}$. We define the *misdoding of the area* given the known subset \hat{A} as:

$$\mu(\hat{A}) = \min_{(r_{e_1}^*, r_{e_2}^*, \dots, r_{e_n}^*) \in \mathcal{R}_e^*} \frac{K(\langle t_{e_1}, t_{e_2}, \dots, t_{e_n} \rangle | \langle t_1, t_2, \dots, t_n \rangle)}{K(\langle t_{e_1}, t_{e_2}, \dots, t_{e_n} \rangle)}$$

References

A good introduction to the study of uncertainties (error analysis in models) in science, and in particular in physics, chemistry, and engineering, is the best-selling text [[taylor2022introduction](#)], which also features the same image of a crashed train than in the introduction to this chapter.

The concept of redundancy has been also investigated in the context of information theory, since we are interested on using codes with low redundancy (see for example [[abramson1963information](#)]).

TODO: Add more references.



14. Nescience

There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know.

Donald Rumsfeld

Following an initial Chapter 9 that established the foundational concepts of entity, representation, and description, and the subsequent Chapters 10, 11, and 12, which introduced novel metrics for miscoding, inaccuracy, and surfeit, we are now equipped with the essential tools to explore the profound concept of nescience in this chapter, focusing on its key characteristics.

Unlike Shannon's entropy or Kolmogorov's complexity, which quantify information, nescience assesses the absence of information, or what remains unknown. The theory of nescience quantifies our ignorance regarding a research entity through the three newly examined metrics: miscoding, inaccuracy, and surfeit. Miscoding evaluates the fidelity of an entity's representation as a sequence of symbols; inaccuracy gauges the precision with which our optimal model captures this symbolic representation; and surfeit examines the depth of our comprehension of the description, measured by the model's length or symbol count. The challenge with these metrics lies in their inherent conflict: improving one may worsen the others. Our goal is to find a method to minimize all these elements simultaneously. This requirement underlines, according to our interpretation, that science is fundamentally a multi-objective optimization problem.

One of the most significant outcomes of our definition of nescience, grounded in the metrics of miscoding, inaccuracy, and surfeit, is its ability to categorize the domain of research topics into two distinct areas. The first area is referred to as the known unknown, which encompasses the topics we recognize as not fully understanding yet acknowledge our lack of complete comprehension. The second area is the unknown unknown, consisting of topics yet to be discovered. A crucial application of the theory of nescience lies in its use as a methodology for uncovering what resides within the unknown unknown.

Another notable consequence of the nescience concept is the highly counterintuitive notion that, for certain topics, further research may prove to be counterproductive. In these instances, the more

research we conduct, the less we know, as it becomes impossible to expand our knowledge beyond a critical threshold, despite being far from possessing a comprehensive understanding.

14.1 Nescience

Intuitively, our understanding of an entity should be based on the quality of the model we use to describe it, namely, its ability to explain why things happen. In the theory of nescience, we propose a quantitative measure of our ignorance about a research entity using the miscoding of a string-based representation of the entity, and the inaccuracy and surfeit of the model describing this representation. Miscoding is significant because it indicates how well the representation encodes the entity, inaccuracy measures how closely our model comes to adequately describing the representation, and surfeit quantifies the extent of unnecessary effort invested in the model. We believe that the objective of Science should be to minimize these three quantities: miscoding, inaccuracy, and surfeit. Unfortunately, these metrics are conflicting, in the sense that decreasing one could increase the other two.

According to the theory of nescience, science aims to address the following multiobjective optimization problem¹:

The Science Problem

$$\begin{aligned} \text{minimize} \quad & \{\mu(r), \iota(d, r), \sigma(d, r)\} \\ \text{subject to} \quad & (r, d) \in \mathcal{B}^* \times \mathcal{D} \end{aligned}$$

A *scientific method*, detailed further in Section 16.2, refers to any algorithm or computable procedure capable of addressing, or providing a closely approximate resolution to, the previously discussed minimization challenge. This encompasses a wide range of techniques and methodologies aimed at systematically reducing the metrics of miscoding, inaccuracy, and surfeit within the realm of scientific inquiry, thereby enhancing the precision and conciseness of our representations and models.

The feasible region is the Cartesian product $\mathcal{B}^* \times \mathcal{D}$ of the set of finite binary strings \mathcal{B}^* and the set of descriptions \mathcal{D} . The decision vectors are pairs (r, d) , called *hypothesis*, composed of a representation and a description, and the objective functions to minimize are miscoding, surfeit, and inaccuracy. The objective region is the subset $\mathbf{Z} \subset \mathbb{R}^3$, and its elements are the objective vectors.

In our characterization of science and the scientific method, we do not involve the set \mathcal{E} of entities, since requiring knowledge of the entity $e \in \mathcal{E}$ under study would render science an ill-defined problem for the majority of research topics. Science is essentially a problem of manipulating strings of symbols. From a practical perspective, science is about finding strings that have a meaningful interpretation in the real world and that enable us to solve practical problems. From a more theoretical perspective, the goal of science would be to understand how an unknown abstract oracle functions.

If the set \mathcal{R}_e of representations of the particular entity e in which we are interested is known, or approximately known, we could restrict the science problem to:

$$\begin{aligned} \text{minimize} \quad & \{\mu(r), \iota(d, r), \sigma(d, r)\} \\ \text{subject to} \quad & (r, d) \in \mathcal{R}_e \times \mathcal{D} \end{aligned}$$

In the theory of nescience, we are mostly interested in the decision space $\mathcal{B}^* \times \mathcal{D}$ of representations and descriptions rather than in the objective space $\mathbf{Z} \subset \mathbb{R}^3$ of metric values. In the following

¹Technically speaking, science is a deterministic discrete nonlinear nonconvex nondifferentiable multiobjective optimization problem with a single decision maker.

definitions, we reintroduce some of the concepts of multiobjective optimization (see Section 7.5) for the particular case of the science problem.

Pareto Optimality

If the representation and description currently in use to describe an entity are not perfect, our goal is to find another representation, or another description, that reduces at least one of the metrics miscoding, inaccuracy, or surfeit without deteriorating either of the other two.

Definition 14.1.1 We say that a decision vector $(r, d) \in \mathcal{B}^* \times \mathcal{D}$ *dominates* another decision vector $(r', d') \in \mathcal{B}^* \times \mathcal{D}$ if it improves in one of the metrics miscoding, inaccuracy, or surfeit without deteriorating either of the other two.

We could discover a new representation that more accurately encodes the entity of interest without diminishing the quality of its descriptive model. Alternatively, we might identify a new description that addresses inaccuracies without augmenting the surfeit, or we could find a more concise description that does not compromise accuracy.

■ **Example 14.1** Consider an experiment where we've gathered several observations r and applied a mathematical model f_1 from a family of models \mathcal{M}_1 , resulting in an inaccuracy of i_1 . Subsequently, we might fit a second model f_2 from an alternative family of models \mathcal{M}_2 . This second model is smaller than i_2 while maintaining the same inaccuracy. In this scenario, we would say that the hypothesis $B = (r, i_2)$ dominates the hypothesis $A = (r, i_1)$ although they are not superior in the individual objective of inaccuracy. ■

For the majority of entities, there does not exist a single solution that simultaneously minimizes the three metrics. Instead, what we have is a collection of Pareto optimal solutions that define an optimal frontier.

Definition 14.1.2 We say that a decision vector $(r, d) \in \mathcal{B}^* \times \mathcal{D}$ is *Pareto optimal* if there does not exist another decision vector $(r', d') \in \mathcal{B}^* \times \mathcal{D}$ such that (r', d') dominates (r, d) . The set of Pareto optimal solutions, denoted by $\mathbf{P}_{\mathcal{B}^* \times \mathcal{D}}$, is called the *Pareto frontier*.

In the realm of scientific research, the concept of the Pareto frontier, as defined by the set of Pareto optimal solutions $\mathbf{P}_{\mathcal{B}^* \times \mathcal{D}}$, plays a crucial role. It delineates the boundary of optimal trade-offs between the conflicting metrics of miscoding, inaccuracy, and surfeit, without one being improved at the expense of the others. This frontier represents the spectrum of best-achievable balances, guiding researchers to identify models and representations that offer the most scientifically rigorous understanding of their subject matter (see Section 14.6). However, in certain cases or specific practical applications, it might be justifiable to adopt a solution that is not Pareto optimal. For example, we might prioritize one metric significantly over others due to its relevance or the specific objectives of the research, accepting a degree of compromise in the non-prioritized metrics (see Section 14.2).

Building on the concept of Pareto optimality, where a solution is considered optimal if no other solution can improve one objective without worsening another, we introduce the notion of weakly Pareto optimality. A decision vector is deemed weakly Pareto optimal if there is no other vector that can improve all objectives simultaneously. It differs from Pareto optimality in that it includes solutions that, although they may not be superior in some individual objective, are not outperformed across all dimensions.

Definition 14.1.3 We say that a decision vector $(r, d) \in \mathcal{B}^* \times \mathcal{D}$ *weakly dominates* another decision vector $(r', d') \in \mathcal{B}^* \times \mathcal{D}$ if it improves the three metrics miscoding, inaccuracy and surfeit at the same time, that is, $\mu(r') < \mu(r)$, $\iota(d', r') < \iota(d, r)$ and $\sigma(d', r') < \sigma(d, r)$.

A decision vector is weakly Pareto optimal if it does not exist another decision vector that

improves over the three metrics of miscoding, inaccuracy and surfeit.

Definition 14.1.4 We say that a decision vector $(r, d) \in \mathcal{B}^* \times \mathcal{D}$ is *weakly Pareto optimal* if there does not exist another decision vector $(r', d') \in \mathcal{B}^* \times \mathcal{D}$ such that (r', d') weakly dominates (r, d) . The set of Pareto optimal solutions, denoted by $\mathbf{P}_{\mathcal{B}^* \times \mathcal{D}}$, is called the *weakly Pareto frontier*.

If a decision vector is weakly Pareto optimal that means we cannot find another decision vector that improves over the three metrics. However, we can still find a decision vector that improves over one of the metrics without deteriorating the other, that is, a decision vector that is Pareto optimal. The Pareto frontier is a subset of the weakly Pareto frontier. In the theory of nescience we will work mostly with the set of Pareto optimal solutions, to the detriment of set of weak Pareto solutions.

■ **Example 14.2** Based on the assumptions of Example 14.1 the hypothesis A still could be weakly Pareto optimal, but it cannot be Pareto optimal, since it is dominated by the hypothesis B, but is not weakly dominated by the hypothesis B. ■

The concept of Pareto and weakly Pareto optimality can be particularized to the case in which the set \mathcal{R}_e of representations of the particular entity e is known.

Range of Solutions

As discussed in Section 7.5.1, an objective vector that achieves the minimum possible value for all objective functions is termed an ideal objective vector. For the science problem, this ideal vector is represented by the origin $(0, 0, 0)$, symbolizing the complete elimination of miscoding, inaccuracy, and surfeit.

Proposition 14.1.1 The ideal objective vector for the science problem is the origin $(0, 0, 0)$.

Proof. Proposition 10.1.1 demonstrated that miscoding is greater than or equal to zero, and Proposition 10.1.2 showed that it can be equal to zero. Similarly, Proposition 11.5.1 established that inaccuracy is greater than or equal to zero, and Proposition 13.1.4 indicated that a zero value is attainable. Finally, Proposition 14.2.1 argued for surfeit being greater than or equal to zero, and Proposition 12.1.2 confirmed that achieving a value of zero is possible. ■

A decision vector $(r, d) \in \mathcal{B}^* \times \mathcal{D}$ is ideal if it possesses zero miscoding, zero inaccuracy, and zero surfeit. This means that the representation r is valid, the model d 's output is r , and no shorter model d' exists that also has zero inaccuracy. Intuitively, a decision vector (r, d) is ideal if there is an entity $e \in \mathcal{E}$ such that r perfectly encodes e , and the model d for r is both accurate and minimal.

Ideal decision vectors embody the notion of perfect knowledge within the theory of nescience. Unfortunately, for most practical applications, achieving the ideal vector is not feasible due to the conflicting nature of the metrics miscoding, inaccuracy, and surfeit.

■ **Example 14.3** TODO: in practice it is impossible to reach the objective vector. ■

The upper bound of the Pareto optimal set is given by the nadir objective vector. In the theory of nescience, the nadir vector would be the $(1, 1, 1)$ vector, in which we have the maximum miscoding, minimal accuracy, and maximum surfeit.

Proposition 14.1.2 The nadir objective vector of the science problem is the vector $(1, 1, 1)$.

Proof. Proposition 10.1.1 demonstrated that miscoding is greater than or equal to zero, and Proposition 10.1.2 showed that it can be equal to zero. Similarly, Proposition 11.5.1 established that inaccuracy is greater than or equal to zero, and Proposition 13.1.4 indicated that a zero value is attainable. Finally, Proposition 14.2.1 argued for surfeit being greater than or equal to zero, and Proposition 12.1.2 confirmed that achieving a value of zero is possible. ■

The nadir vector represents the situation in which we have a decision vector (r, d) with zero knowledge, that is, our representation r does not contain symbols related to the entity e under study, the description d produces a string completely different from r , and it has a maximum length.

■ **Example 14.4** TODO: in practice it is impossible to reach the nadir objective vector. ■

Trade-offs

In Section 7.5.2 we saw that since the functions we want to minimize are conflicting, we have to assume that the only way to gain a benefit in one aspect of the problem is to lose something in another aspect, what it is called a trade-off. In this section we study the different tradeoffs that can arise in the theory of nescience.

TODO: introduce the concept of partial miscoding - inaccuracy

Definition 14.1.5 Let $(r_1, d), (r_2, d) \in \mathcal{B}^* \times \mathcal{D}$ be two decision vectors. We define the partial ratio of change between the functions miscoding and inaccuracy for the vectors $(r_1, d), (r_2, d)$ as:

$$\Delta_{\mu i}((r_1, d), (r_2, d)) = \frac{\iota(r_1, d) - \iota(r_2, d)}{\mu(r_1) - \mu(r_2)}$$

TODO: study its properties

TODO: introduce the concept of total miscoding - inaccuracy

Definition 14.1.6 Let $(r_1, d_1), (r_2, d_2) \in \mathcal{B}^* \times \mathcal{D}$ be two decision vectors. We define the total ratio of change between the functions miscoding and inaccuracy for the vectors $(r_1, d_1), (r_2, d_2)$ as:

$$\Delta_{\mu \sigma}((r_1, d_1), (r_2, d_2)) = \frac{\iota(r_1, d_1) - \iota(r_2, d_2)}{\mu(r_1) - \mu(r_2)}$$

TODO: study its properties

TODO: introduce the concept of partial miscoding - surfeit

Definition 14.1.7 Let $(r_1, d), (r_2, d) \in \mathcal{B}^* \times \mathcal{D}$ be two decision vectors. We define the partial ratio of change between the functions miscoding and surfeit for the vectors $(r_1, d), (r_2, d)$ as:

$$\Delta_{\mu \sigma}((r_1, d), (r_2, d)) = \frac{\sigma(r_1, d) - \sigma(r_2, d)}{\mu(r_1) - \mu(r_2)}$$

TODO: study its properties

TODO: introduce the concept of total miscoding - surfeit

Definition 14.1.8 Let $(r_1, d_1), (r_2, d_2) \in \mathcal{B}^* \times \mathcal{D}$ be two decision vectors. We define the total ratio of change between the functions miscoding and surfeit for the vectors $(r_1, d_1), (r_2, d_2)$ as:

$$\Delta_{\mu \sigma}((r_1, d_1), (r_2, d_2)) = \frac{\sigma(r_1, d_1) - \sigma(r_2, d_2)}{\mu(r_1) - \mu(r_2)}$$

TODO: study its properties

TODO: introduce the concept of partial inaccuracy - surfeit

Definition 14.1.9 Let $(r, d_1), (r, d_2) \in \mathcal{B}^* \times \mathcal{D}$ be two decision vectors. We define the partial

ratio of change between the functions inaccuracy and surfeit for the vectors $(r, d_1), (r, d_2)$ as:

$$\Delta_{\iota\sigma}((r, d_1), (r, d_2)) = \frac{\iota(r, d_1) - \iota(r, d_2)}{\sigma(r, d_1) - \sigma(r, d_2)}$$

TODO: study its properties

TODO: introduce the concept of total inaccuracy - surfeit

Definition 14.1.10 Let $(r_1, d_1), (r_2, d_2) \in \mathcal{B}^* \times \mathcal{D}$ be two decision vectors. We define the total ratio of change between the functions inaccuracy and surfeit for the vectors $(r_1, d_1), (r_2, d_2)$ as:

$$\Delta_{\mu\sigma}((r_1, d_1), (r_2, d_2)) = \frac{\iota(r_1, d_1) - \iota(r_2, d_2)}{\sigma(r_1, d_1) - \sigma(r_2, d_2)}$$

TODO: study its properties

TODO: study the concept of properly Pareto optimal solutions in the context of the theory of nescience.

14.2 Minimizing Nescience

From a mathematical point of view, any of the solutions that compose the Pareto set would be a valid solution to the Science Problem. In fact, the problem is considered to be solved when all the solutions of the Pareto optimal set are found. However, in science, this is rarely enough, and we prefer to have a single solution. The goal of the decision maker is to provide an ordering for the optimal solutions, or equivalently, add a preference of one solution over the others.

Introduce the concept of nescience decision maker as a function of miscoding, inaccuracy and surfeit, value function

Definition 14.2.1 Let $(r, d) \in \mathcal{B}^* \times \mathcal{D}$ be a decision vector. A decision maker for the science problem is a multivariate function $f : \mathbb{R}^3 \rightarrow \mathbb{R}$, that assigns to each triplet $(\mu(r), \iota(d, r), \sigma(d, r))$ the real value $f(\mu(r), \iota(d, r), \sigma(d, r))$.

The good news are that we have designed those metrics so that they are commensurable, that is, all of them are measured using the same unit: the length of a computer program. Moreover, they are in the same scale, since they have been normalized to the interval between zero and one.

14.2.1 Global Criterion

The global criterion (see Section 7.5.3) is a method for solving multi-objective optimization problems in which the distance between some reference point and the feasible objective region is minimized. As reference point it is usually used the ideal vector, which in our case is the origin $(0, 0, 0)$ of the objective space. And as distance, we could use different metrics. For example, the global criterion based on the origin and an Euclidean distance requires to solve the following minimization problem:

$$\begin{aligned} & \text{minimize} && \sqrt{\mu(r)^2 + \iota(d, r)^2 + \sigma(d, r)^2} \\ & \text{subject to} && (r, d) \in \mathcal{B}^* \times \mathcal{D} \end{aligned}$$

As Proposition 7.5.1 proved, the solutions provided by the global criterion are Pareto optimal. However, not all the solutions from the Pareto frontier are considered as candidate solutions using this method.

We do not need to normalize the candidate solutions before to solve the minimization problem since the metrics we are using are already normalized.

Some examples of alternative distances than can be used with the global criterion method are the following:

TODO: Filter out non-distances.

- Arithmetic mean: $\frac{\mu(r) + \iota(d,r) + \sigma(d,r)}{3}$
- Geometric mean: $(\mu(r) \times \iota(d,r) \times \sigma(d,r))^{1/3}$
- Product: $\mu(r) \times \iota(d,r) \times \sigma(d,r)$
- Addition: $\mu(r) + \iota(d,r) + \sigma(d,r)$
- Harmonic mean: $\frac{3}{\frac{1}{\mu(r)} + \frac{1}{\iota(d,r)} + \frac{1}{\sigma(d,r)}}$

TODO: Mention advantages and drawbacks of this alternative solutions. Geometric mean, product and harmonic mean have the problem that the nescience is zero, or not defined, if one of the three metrics (mCoding, inaccuracy or surfeit) is zero.

■ **Example 14.5 TODO: update this example** Let $t \in \mathcal{T}$ a topic, and d_1 and d_2 two descriptions of t . Assume that the error of d_1 is 0.1 and its redundancy 0.4, and that d_2 has an error 0.2 and a redundancy of 0.2. According to Definition ??, the nescience of topic t given the description d_1 would be 0.41, and the nescience given the description d_2 would be 0.28. Our unknown about topic t would be smaller in case of description d_2 than with description d_1 . ■

TODO: rewrite this paragraph What Example 14.5 show us is that, given our definition of nescience, we should prefer a less redundant explanation that maybe does not describe perfectly a topic, to a highly redundant description that fits the topic much better. The Occam's razor principle states that between two indifferent alternatives we should select the simplest one. Our theory states that even in case they are not indifferent alternatives, it might be worth to select the the simplest one. The theory of nescience has not been designed to find the true about a topic, in its philosophical sense, but to find the best theory that can be used in practice. In this sense, we borrow concepts and ideas from the area of machine learning, and in particular, from the problem of *model overfitting* when dealing with the results of an experiment (see Chapter ?? for more information about this problem).

Proposition 14.2.1 We have that $0 \leq \rho(d_t) \leq 1$ for all $t \in \mathcal{T}$ and $d_t \in \mathcal{D}$.

Proof. Given that $l(d_t) > 0$ and that $K(t) > 0$, since they are the lengths of non-empty strings, we only need to prove that $l(d_t) \geq K(t)$; however $l(d_t) < K(t)$ is a contradiction with the fact that $K(t)$ is the length of the shortest possible Turing machine that prints out t . ■

14.2.2 Weighting Method

14.2.3 The other method

14.2.4 Evolutionary Methods

14.3 Joint Nescience

In Section 9.4 we introduced the concept of joint representation as the concatenation of two or more strings, and in Section 10.2 we studied how mCoding is affected when we concatenate these representations. In this section we are going to see how nescience, as a global metric, is affected when joining representations.

Definition 14.3.1 Let $s, t \in \mathcal{B}^*$ be two representations, and $d \in \mathcal{D}$ a description. We call *joint nescience*, denoted by $\mathcal{N}(st, d)$, to the nescience of the joint representation st and the description d .

We are interested to know how the joint nescience of $\mathcal{N}(st, d)$ is related to the individual nesciences $\mathcal{N}(s, d)$ and $\mathcal{N}(t, d)$.

TODO: rewrite paragraph. As we have seen in Section 9.4 there is no way to guarantee that the mCoding of a joint representation will be smaller, or greater, than the mCoding of the individual

representation, not even in case that s and t encode the same entity e . In the same way, we saw in Section XXX that the inaccuracy of a fixed model d for a joint representation st could be greater, or smaller, than the inaccuracy of the model for the individual representations s or t . And finally, in case of surfeit, as we saw ...

TODO: Is there anything we can say about joint nescience? Conumtative? Extend to the concatenation of multiple representations.

14.4 Conditional Nescience

TODO: Introduce the section, what we are aiming for.

TODO: Introduce the concept of conditional nescience

Definition 14.4.1 Let $(r, d) \in \mathcal{B}^* \times \mathcal{D}$ be a topic, and $s \in \mathcal{B}^*$ be an arbitrary string. We define the *conditional nescience* of the topic (r, d) given the string s , denoted $\mathcal{N}(r, d_{r|s})$, as the nescience of the representation r and the conditional description $d_{r|s}$.

TODO: Explain the intuition behind this concept.

TODO: Provide a practical example.

TODO: Provide a maximum and a minimum for the concept

The next proposition states that the nescience of a topic can only decrease if we assume the background knowledge given by another topic. : **Explain the intuition behind this property.**

TODO: Prove something like $N(t, s) = N(t) + N(s|t)$

TODO: Check the following proposition

Proposition 14.4.1 $N_{t|s} = N_{s|t}$ if, and only if, $N_{t|s} = N_{s|t} = 0$.

Finally, next proposition states the relation between nescience, conditional nescience and joint nescience.

TODO: review

Proposition 14.4.2 Given any two topics $t, s \in T$, we have that $N_{t|s} \leq N_t \leq N_{(s,t)}$.

Proof. **TODO** ■

The concept of conditional nescience can be extended to multiple topics s_1, \dots, s_n , using the standard encoded string $\langle s_1^*, \dots, s_n^*, a \rangle$ and the multiple conditional complexity $K(t | s_1, \dots, s_n)$.

Definition 14.4.2 Let $t, s_1, \dots, s_n \in T$ a collection of topics. The *conditional nescience* of topic t given the topics s_1, \dots, s_n and current best description \hat{d}_t is defined as:

TODO

14.5 Nescience of Areas

TODO: Rewrite this section.

Topics can be grouped into research areas. The concept of area is useful as long as all the topics included in the area share a common property. The particular details of the grouping properties depend on the practical applications of the theory.

Definition 14.5.1 An area A is a subset of topics, that is $A \subset \mathcal{T}$.

Areas can overlap, that is, given two areas A and B it might happen that $A \cap B \neq \emptyset$. Areas can be subsets of other areas, creating an hierarchy of areas.

■ **Example 14.6** If the set of topics is "mathematics", examples of areas could be "calculus", "geometry" or "algebra". The areas "probability" and "statistics" largely overlap. The area "Bayesian inference" is a subarea of the area "probability". ■

TODO: study the properties of the research areas.

In the same way we studied the properties of individual topics, we could study the properties of areas. An *area* is a subset of topics $A \subset T$. The concept of area is useful as long as all the topics included in the area share a common property. What is exactly that property they share depends on the particular set T .

Definition 14.5.2 Given an area $A \subset T$, we define the *average complexity of the area* C_A as $C_A = \frac{1}{n} \sum_{t \in A} C_t$, and the *average nescience of the area* N_A as $N_A = \frac{1}{n} \sum_{t \in A} N_t$, where n is the cardinality of A .

For example, in case of research topics, an area could be a knowledge area, like biology, that will contain all the topics classified under that area. In this way we could compute and compare the complexity (how difficult is to understand) and the nescience (how ignorant we are) of mathematics, physics, biology, social sciences, and other disciplines.

TODO: This definition can only be introduced once we have the concept of best current description.

An even easier approximation of the concept of redundancy of an area, is based on the average redundancy of the topics that compose that area.

Definition 14.5.3 Let $A \in \mathcal{T}$ an area, and d_A a description. We define the *average redundancy of the description* d_A as:

$$\bar{\rho}(d_A) = \sum$$

TODO: Study some properties of this definition

TODO: How these three definitions relate to each other?

14.6 Perfect Knowledge

TODO: Rewrite this section

As we have said, in our opinion, the objective of scientific research should be to reduce the nescience of topics as much as possible. When it is not possible to reduce more the nescience of a topic, we propose to say we have reached a perfect knowledge. A consequence of

Definition 14.6.1 — Perfect Knowledge. If the nescience of a topic t is equal to zero ($v(t) = 0$), we say that we have reached a *perfect knowledge* about topic t .

If $v(t) = 0$ we have that $\varepsilon(t) = \rho(t) = 0$, that is, perfect knowledge is achieved when we can fully reconstruct the original topic given its description, and the description does not contain any redundant elements. A consequence of our definition is that perfect knowledge implies randomness, that is, incompressible descriptions. The converse, in general, does not hold. The common point of view is that a random string should make nonsense, since this is what randomness is all about. However, in the theory of nescience, by random description we mean a description that contains the maximum amount of information in the less space as possible (it contains no redundant elements).

■ **Example 14.7** Aristotelian physics is an inaccurate description of our world, since it makes some predictions that do not hold in reality (for example, planets do not orbit around the earth). We could use a description of the Aristotelian physics and compress it using a standard compression program. The compressed file would be a random description (zero redundancy). However, given

that description, our nescience would not be zero, that is, our knowledge would not be perfect, since the error of the description is not zero. ■

TODO: Show how nescience evolves with time

TODO: Define the concept of weak nescience

TODO: Explain how weak nescience and nescience relates to each other

TODO: Show how the weak nescience converges to nescience in the limit

Theorem 14.6.1 Let $t \in T$ a topic, and $\{d_1, d_2, \dots\}$ where $d_i \in D_t$ a set of descriptions such that $l(d_i) < l(d_j)$ for each $i < j$, then

$$\lim_{x \rightarrow \infty} \hat{N}_i = N_t$$

Proof. To Be Done ■

■ **Example 14.8** In order to clarify how the above theorem can be applied in practice, in Figure 14.1 it is shown an hypothetical example of the typical research process required to understand a scientific topic $t \in T$. In time t_1 we have a description with length 12, whose compressed version has a length of 5, and so, its nescience is 1.4. In time t_2 , supposedly after some intense research effort, our current description length has been reduced to 8 with a complexity of 4, and our nescience has decreased to 1. In the limit, the description length will be equal to its complexity (incompressible description), and the nescience will be 0. In this moment we could say that we have a *perfect knowledge* about that particular research topic. ■

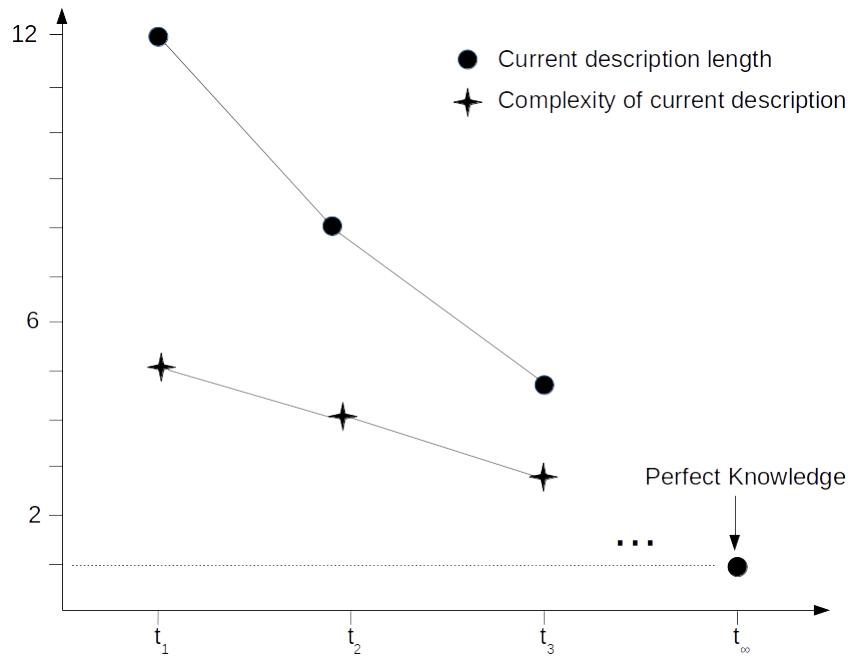


Figure 14.1: In Pursuit of Perfect Knowledge

TODO: Introduce the following definition.

Definition 14.6.2 Let $s_1, \dots, s_n \in T$ a collection of topics. The *joint nescience* of topics topics s_1, \dots, s_n given the current best descriptions $\hat{d}_{s_1}, \dots, \hat{d}_{s_n}$ is defined as:

TODO

TODO: Mention, or prove, the properties of the generalized nescience

14.7 Current Best Description

TODO: Rewrite this section

We are also interested in our current understanding of the concatenation of two topics.

Definition 14.7.1 Given $t, s \in \mathcal{T}$ two different topics, and the set $\mathcal{D}_{t,s} = \{d \in \mathcal{B}^*, d = \langle TM, a \rangle : TM(a) = \langle t, s \rangle\}$, let $\hat{d}_{t,s}$ be a distinguished element of $\mathcal{D}_{t,s}$. We call $\hat{d}_{t,s}$ our *current best joint description* of t and s .

The concept of best joint description could be a little bit misleading, since given that the concatenation of any two topics is another topic, we could ask ourselves why do not simply use our current best description of the topic represented by that concatenation. That would make sense only in case that somebody has already studied both topics together. However, for the overwhelming majority of the possible combinations of topics, nobody has studied them yet.

■ **Example 14.9** Let t and s two different topics, and assume that nobody has studied them together before. In this case, our current best description $\hat{d}_{t,s}$ would be $\langle TM, \langle \hat{d}_t, \hat{d}_s \rangle \rangle$, where TM is a Turing machine that given the input $\langle \hat{d}_t, \hat{d}_s \rangle$ prints out the string ts . If $\hat{d}_t = \langle TM_t, a_t \rangle$ and $\hat{d}_s = \langle TM_s, a_s \rangle$, the machine TM will decode \hat{d}_t , run $TM_t(a_t)$ to print out t ; then it would do the same for \hat{d}_s to print s . ■

As we said in the preface of this chapter, in order to compute how much we do not know about a topic, first we need a way to quantify what we already know about that topic. How much we know about a topic will be given by our current best known description of that topic.

Definition 14.7.2 Given the set of descriptions \mathcal{D}_t of a topic $t \in \mathcal{T}$, let \hat{d}_t be a distinguished element of \mathcal{D}_t . We call \hat{d}_t our *current best description* of t .

Which description is the current best description is something that depends on our current knowledge about the particular area in which the theory of nescience is being applied.

14.8 Nescience based on Datasets

TODO: Perhaps we should remove this section

Some topics can be described using a mathematical model that can be evaluated by a computer. For those topics we could estimate their complexity using a sample dataset, for example the result of an experiment. This property allows us, among other things, to compare how well topics are described by different models (see Chapter ??).

Definition 14.8.1 Let $t \in \mathcal{T}$ be a topic, $D = \{x_1, x_2, \dots, x_n\}$ the result of running an experiment that describes t , and H a mathematical hypothesis for t . The complexity of the topic t given the current description H and the dataset D is given by

$$\hat{C}_t = L(H) + L(D | H)$$

as it is described by the minimum description length principle.

Given the dataset D and the model H we can compute our current nescience of a topic as it is described by the next definition.

Definition 14.8.2 Let $t \in T$ a topic, $D = \{x_1, x_2, \dots, x_n\}$ the result of running an experiment that describes t , C a code that minimizes the length of D , and H a mathematical hypothesis for t . The current complexity of the topic t given the dataset D and the hypothesis H is given by

$$N_t = \frac{\hat{C}_t - l_C(D)}{l_C(D)}$$

In practice, the dataset D also allows us to approximate our current nescience of a topic t , given the model H . What we have to do is to use a near minimal encoding for D , for example, by using a Huffam encoding.

14.9 Unknown Unknown

TODO: pending

Known unknown

Unknown unknown

Finally, there exists a last category of unknown, what we call the unknowable unknown unknown. This category refers to those entities we

Unknowable unknown unknown. In this book we are interested in the unknown unknown

References

TODO: Add the paper of Chaitin about the Berry paradox

TODO: That there are numbers that are not computable can be found in the original paper of Turing

TODO: Perhaps I should provide a couple of references in epistemology and ontology



15. Interesting Questions

*It is not the answer that enlightens,
but the question.*
Eugène Ionesco

In this chapter, we propose a set of metrics for classifying research topics based on their potential as a source of interesting problems, along with a methodology for assisted discovery of new questions. The methodology can be used to identify new applications of existing tools to solve open problems and to discover new, previously unexplored research topics. The methodology is applicable to both intradisciplinary and interdisciplinary topics, but the most interesting outcomes are obtained in the latter case. In Chapters 19 and 20, we demonstrate the application of the methodology in practice and suggest new questions and research topics.

Our primary assumption is that a research question is interesting if it meets the following three criteria:

- C1** The question should be new and original, meaning that it has not been previously considered.
- C2** Upon its resolution, there should be a significant increase in our knowledge about one or more specific research topics.
- C3** It should have practical applications that could have a substantial (hopefully positive) impact on people's lives.

Some researchers may argue that the requirements presented may not be the most suitable. For instance, researchers from the so-called "hard sciences", such as pure mathematics and theoretical physics, might object that practical applications are not a crucial factor in pursuing an interesting open problem. In such situations, the metrics introduced can be redefined since they are simply mathematical abstractions, and the same methods can still be applied. The methodology described is universal and can be employed in various domains, not only in discovering new research questions. In fact, the metrics and methods outlined can be utilized in any field where there is a vast collection of interconnected describable objects, and the objective is to uncover new and previously unknown

objects. The precise definition of concepts like relevance graph, applicability graph, or maturity will be contingent on the field in which the approach is being employed. Nevertheless, as in the case of Chapter 14, we prefer to present the methodology and new concepts in the specific context of scientific research because it aids in their comprehension.

It is worth mentioning the relationship between this new methodology and the areas of computational creativity and artificial intelligence. What is proposed in this chapter is an algebraic approach to the assisted discovery of potentially interesting questions. The intention is not for the computer to understand the meaning of the questions posed. Furthermore, not all the questions generated are necessarily relevant or even meaningful. It is the responsibility of the researchers to assess the proposed combinations of topics and determine if any of the questions are appropriate.

We have already explored two dimensions for classifying topics: miscoding (Chapter 10) and mismodel (Chapter 12). These metrics enable us to quantitatively assess our understanding of a topic, which we refer to as nescience. According to criterion **C2** of our list of requirements for questions, the higher the nescience of a topic, the greater its potential as a source of interesting research questions. In this section, we will introduce two additional metrics for characterizing topics: relevance and applicability. Relevance measures the impact a topic has on people's lives and serves as a complement to nescience. Applicability measures how frequently a topic has been applied in other areas and enables us to identify new applications of already existing technologies.

15.1 Relevance

Before to measure relevance of a topic, that is, its impact in people's life, we have to introduce the concept of *relevance graph*. The relevance graph is a graph that describes which people is affected by which research topics.

Definition 15.1.1 We define the *relevance graph*, denoted by **RG**, as the bipartite graph $\mathbf{RG} = (\mathcal{T}, \mathcal{P}, E)$, where \mathcal{T} is the set of topics, \mathcal{P} the set of people, and $E \subseteq \{(i, j) : i \in \mathcal{T}, j \in \mathcal{P}\}$ is the set of arcs between topics and people. An arc (i, j) belong to E if, and only if, person j is affected by topic i .

When we refer to the set of people \mathcal{P} , we are referring to all individuals in the world. An edge in the relevance graph indicates that someone is affected by a topic, rather than being interested in it. The exact meaning of "being affected by" is a difficult to define concept that is beyond the scope of this book. Therefore, our definition of the relevance graph is a mathematical abstraction. In Section 19.2, we will explore how to approximate this quantity in the case of scientific research topics. In Chapter 18, we will provide an alternative interpretation of the relevance graph in the context of measuring software quality.

■ **Example 15.1** A man who is affected by ALS (amyotrophic lateral sclerosis) disease will be connected to the ALS topic in the graph. His spouse will also be connected because she is likely to be impacted by the consequences of the disease as well. However, a researcher who is interested in ALS as a research problem, rather than someone suffering the disease, will not be connected to the ALS node in the graph. ■

Optionally, we can add a weight $w_{ij} \in [0, 1]$ to the edges of the graph to specify the degree in which a person j is affected by a topic i . A weight of 1 could represent a life-or-death dependence, and 0 would mean that this person is not affected at all. In Figure 15.1 it is depicted an example of relevance graph.

Definition 15.1.2 We define the *relevance* of a topic $t \in \mathcal{T}$, denoted by $R(t)$, as the degree of the node t in the relevance graph, that is, $R(t) = \deg(t)$.

Intuitively, the higher the relevance of a topic, the higher its potential as a source of interesting

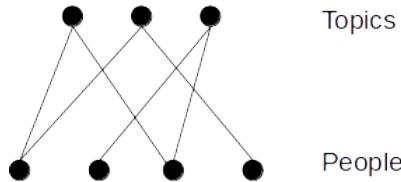


Figure 15.1: Relevance Graph

questions, since we will be working on a problem that affects many people.

Sometimes it is convenient to work with a normalized version of the relevance of a topic.

Definition 15.1.3 We define the *normalized relevance* of a topic $t \in \mathcal{T}$, denoted by $\bar{R}(t)$, as the normalized degree of the node t in the relevance graph, that is, $\bar{R}(t) = \deg(t)/d(E)$.

We could have computed also $\deg(p)$, that is, the number of edges that links to a person p in the relevance graph, as a measure of the number of topics that affects a particular person. However, this quantity is not used in the theory of nescience. The relation between $\deg(t)$ and $\deg(p)$ is given by the degree sum formula:

$$\sum_{t \in \mathcal{T}} \deg(t) = \sum_{p \in \mathcal{P}} \deg(p) = d(E)$$

Next proposition proves that adding more topics to a research project can only increase its relevance. Of course, a research project dealing with "life, the universe and everything" would be a highly relevant one, but very impractical as well. How to properly combine research topics will be described in Section 15.4.

Proposition 15.1.1 Given any two topics $t_1, t_2 \in \mathcal{T}$, we have that $R(t_1) + R(t_2) \geq R(t_1)$.

Proof. Let S_1 the set of people connected to topic t_1 in the relevance graph, and S_2 the set of people connected to topic t_2 . Since $d(S_1 \cup S_2) = d(S_1) + d(S_2) - d(S_1 \cap S_2)$, and $d(S_2) - d(S_1 \cap S_2) \geq 0$ we have that $d(S_1 \cup S_2) \geq d(S_1)$ and thus $R(t_1) + R(t_2) \geq R(t_1)$. ■

Finally, we define the concept of interestingness of a topic as a source of interesting problems, that is, how likely is that the topic can be used in a new interesting research question, as a function of its nescience and relevance. We can visualize topics in a two-dimensional vector space, in which one dimension is nescience, and the other one is relevance. In this interpretation, a natural choice of interestingness function would be the euclidean distance from the origin.

Definition 15.1.4 Given a topic $t \in \mathcal{T}$, we define the *interestingness of the topic as a problem*, denoted by $IP(t)$, as:

$$IP(t) = \sqrt{v(t)^2 + R(t)^2}$$

Intuitively, a topic is interesting as a problem worth investigating if it has a large relevance (it has high impact in people's life) and a large nescience (it is not very well understood). In this sense, we are borrowing ideas from Popper's falsificationism: the more risky is a conjecture, the higher the advance achieved in science given its confirmation.

■ **Example 15.2** The fixed point theorem has some relevance, since people life's can be indirectly affected by its implications, but since it is a very well understood theorem (our nescience is very low), it is not a very interesting research problem by itself.

World War I is a very relevant topic, because it had a huge impact on many people's life, and also it is not very well understood topic, since it takes hundreds of pages to explain its causes, and there is no general agreement among the specialists. So, according to our definition, it is a very interesting research problem. ■

In practice, it is convenient to work with the normalized version of the relevance metric, so that we can avoid the case that our questions are always related to a small subset of extremely interesting topics.

$$IP(t) = \sqrt{v(t)^2 + \bar{R}(t)^2}$$

15.2 Applicability

As mentioned in Example 15.2, the fixed point theorem is not particularly compelling as a research problem on its own. However, it is an essential mathematical result since it has broad applications in proving many other theorems. In this section, we introduce the concept of *applicability*, which is a novel measure enabling us to identify which topics are important since they can serve as tools for understanding other topics. From a mathematical point of view, we say that a tool can be applied to a topic if the conditional nescience (see Section 14.4) of the topic given the tool is smaller than the unconditional nescience of the topic.

Before to formally define the concept of applicability, first we have to introduce a new bipartite graph that describes which topics has been applied as tools to other topics.

Definition 15.2.1 We define the *applicability graph*, denoted by AG , as the directed graph $AG = (\mathcal{T}, E)$, where \mathcal{T} is the set of research topics, and $E \subseteq \{(i, j) : i, j \in \mathcal{T}\}$. An arc (i, j) belong to E if $N(t_i | t_j) < N(t_i)$. The weight of the arc (i, j) is given by $w_{ij} = N(t_i) - N(t_i | t_j)$.

The applicability graph helps us identify those topics that can be potentially used as tools to understand other topics. The following defintion formally introduces this idea.

Definition 15.2.2 Given the applicability graph $AG = (\mathcal{T}, E)$, the *applicability* of a topic $t_i \in \mathcal{T}$, denoted by $A(t_i)$, is defined as the sum of the weights of the arcs in the outdegree of t_i , that is:

$$A(t_i) = \sum_{(i,j) \in E} w_{ij}$$

where E is the set of arcs in the applicability graph, and w_{ij} is the weight of the arc (i, j) .

A topic with higher applicability will have a greater overall impact on the understanding of other topics, and intuitively, the higher the applicability of a topic, the higher its potential as a tool that can be applied to solve open problems. If a tool has been successfully applied multiple times in the past to address open problems, it is more likely that it can be effectively used to solve other open problems as well.

Next proposition proves that the combination of two topics can only increase their applicability. That is, the more tools we have at our disposal, the more problems we could solve in principle.

Proposition 15.2.1 Given any two topics $t_1, t_2 \in \mathcal{T}$, we have that $A(t_1) + A(t_2) \geq A(t_1)$.

Proof. Use the same argument than in Proposition 15.1.1. ■

In order to better compare the applicability of different topics and understand their relative importance in a standardized way, it is useful to introduce a normalized version of the applicability measure.

Definition 15.2.3 Given the applicability graph $AG = (\mathcal{T}, E)$, the *normalized applicability* of a topic $t_i \in \mathcal{T}$, denoted by $A_n(t_i)$, is defined as the ratio of the applicability of t_i to the maximum possible applicability value, expressed as:

$$\tilde{A}(t_i) = \frac{A(t_i)}{\max_{t_k \in \mathcal{T}} A(t_k)}$$

where $A(t_i)$ is the applicability of topic t_i as defined in Definition 15.2.2, and $\max_{t_k \in \mathcal{T}} A(t_k)$ is the maximum applicability value among all topics in \mathcal{T} .

Normalized applicability scales the original applicability value of a topic to a range between 0 and 1, allowing for easier comparison across different topics.

In practice, it is very difficult to compute the applicability graph, since for the majority of the topics, the quantity $N(t_i | t_j)$ has not been estimated. In order to solve this problem, we can use an approximation to the applicability graph, in which the arcs have no weight, and an edge (i, j) represents that the topic j has been applied to solve the problem i . For example, there would be a direct link between the topics "graph theory" and "recommendation engines", since graph theory has been successfully applied to the problem of how to recommend purchase items to customers over Internet.

Definition 15.2.4 We define the *simplified applicability graph*, denoted by SAG , as the directed graph $SAG = (\mathcal{T}, E)$, where \mathcal{T} is the set of research topics, and $E \subseteq \{(i, j) : i, j \in \mathcal{T}\}$. An arc (i, j) belongs to E if the topic j has been used to understand topic i .

Using this simplified version of the applicability graph, the concept of applicability becomes:

Definition 15.2.5 We define the *simplified applicability* of a topic $t \in \mathcal{T}$, denoted by $SA(t)$, as the outdegree of that node in the applicability graph, that is:

$$SA(t) = \text{outdeg}(t)$$

And the corresponding normalized version becomes:

Definition 15.2.6 Given the simplified applicability graph $SAG = (\mathcal{T}, E)$, the *simplified normalized applicability* of a topic $t_i \in \mathcal{T}$, denoted by $SA_n(t_i)$, is defined as the ratio of the simplified applicability of t_i to the maximum possible simplified applicability value, expressed as:

$$SA_n(t_i) = \frac{SA(t_i)}{\max_{t_k \in \mathcal{T}} SA(t_k)}$$

where $SA(t_i)$ is the simplified applicability of topic t_i as defined in Definition 15.2.2, and $\max_{t_k \in \mathcal{T}} SA(t_k)$ is the maximum applicability value among all topics in \mathcal{T} .

When using topics as tools, we are primarily interested in those topics that are better understood. Generally, relying on a background knowledge that is poorly understood is not advisable, even if it significantly reduces the (conditional) nescience of our problem. In the following definition, we will introduce the concept of the maturity of a topic as one minus its nescience.

Definition 15.2.7 Given a topic $t \in \mathcal{T}$, we define the *maturity* of topic t , denoted as $M(t)$, as:

$$M(t) = v(t)^{-1}$$

Intuitively, the more mature a topic is, the greater its utility in solving other open problems. Highly immature topics should not be applied as tools to address open problems, as doing so would merely transfer our lack of understanding from one topic to another.

■ **Example 15.3** Linear regression is a highly mature topic, since its nescience is very small. ■

Finally, we define the concept of interestingness of a topic as a source of interesting tools, that is, how likely is that the topic can be used to solve a new problem, as a function of its maturity and applicability. Similar to the definition of relevance introduced in the previous section, topics can be visualized within a two-dimensional vector space, where one dimension represents maturity and the other represents applicability. The interestingness of a topic will be determined by its Euclidean distance from the origin.

Definition 15.2.8 Given a topic $t \in \mathcal{T}$, we define the *interestingness of the topic as a tool*, denoted by $IT(t)$, as:

$$IT(t) = \sqrt{M(t)^2 + A(t)^2}$$

Intuitively, a topic is considered interesting as a tool if it is thoroughly understood and has already been applied to numerous other problems.

■ **Example 15.4** The Pythagorean theorem (in a right-angled triangle, the square of the length of the hypotenuse is equal to the sum of the squares of the other two sides) is undoubtedly one of the most widely used and applied theorems in various fields and practical situations, including but not limited to: engineering (calculating distances, angles, and forces in structures and mechanical systems), architecture (determining lengths and angles in building design and construction projects), land surveying (measuring distances and calculating areas of land parcels), physics (analyzing problems in mechanics, optics, and electromagnetism), computer graphics and game development (calculating distances and angles in 2D and 3D spaces) or trigonometry(serving as a foundation for the study of trigonometric functions and their applications). The widespread application of the Pythagorean theorem to so many fields is due to its simplicity and its ability to describe complex geometric relationships. ■

15.3 Interesting Questions

In the quest to uncover novel research questions, combining existing topics from various fields can yield fascinating results. By applying tools and methodologies from one topic to solve problems in another, researchers can explore innovative avenues and potentially make groundbreaking discoveries. In the following section, we delve into the methodology of combining topics to generate questions, along with the concept of interestingness and interdisciplinary research.

In our methodology, a interesting question emerges from the combination of two pre-existing topics. Given a pair of a topics in $t_1, t_2 \in \mathcal{T}$, the question can be framed as "*can we apply the tool described by topic t_1 to solve the problem described by topic t_2 ?*".

Definition 15.3.1 Given two topics $t_1, t_2 \in \mathcal{T}$, a *question*, denoted by $Q_{t_1 \rightarrow t_2}$, is the ordered pair (t_1, t_2) .

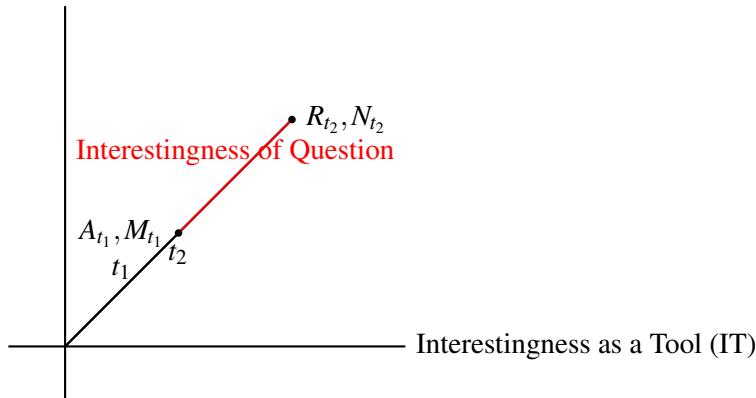
The most intriguing questions emerge when topic t_1 exhibits high interestingness as a tool, and topic t_2 demonstrates high interestingness as a problem. We define the interestingness of a question using the Euclidean distance, taking into account the interestingness of topics t_1 and t_2 as points in a two-dimensional space, with coordinates (A_{t_1}, M_{t_1}) and (R_{t_2}, N_{t_2}) , respectively. The coordinates of the resulting vector represent the interestingness of a question with t_1 as a tool and t_2 as a problem.

Definition 15.3.2 Let t_1 and $t_2 \in \mathcal{T}$ be two topics. The *interestingness* of the question $Q_{t_1 \rightarrow t_2}$, denoted by $IQ_{t_1 \rightarrow t_2}$, is given by:

$$IQ_{t_1 \rightarrow t_2} = \sqrt{(A_{t_1} + R_{t_2})^2 + (M_{t_1} + N_{t_2})^2}$$

Employing the Euclidean distance in this way enables a geometric interpretation of the interestingness of a question based on the vector representation of topics in the interestingness space. The larger the magnitude of the vector representation, the higher the interestingness of the resulting question.

Interestingness as a Problem (IP)



In practice, we must calculate all possible combinations of topics with high interestingness as tools and those with high interestingness as problems. We then select the combinations with the highest interestingness as questions. Naturally, most questions generated using this approach will be meaningless, much like those arising during brainstorming sessions when researchers attempt to identify new tools for tackling difficult problems.

This methodology can be applied in other scenarios as well. For instance, a researcher familiar with problem p might be interested in finding applicable tools to solve it. Similarly, a researcher specializing in tool t may be interested in discovering open problems where his expertise can be applied.

The above procedure can be easily generalized to encompass multiple tools and possibly multiple problems. This leads to the application of two tools to a given problem ($t_1 + t_2 \rightarrow p$), the application of a single tool to the combination of two problems ($t \rightarrow p_1 + p_2$), and so on. The exact meaning of these tool and problem combinations depends on the topics themselves and is left to the researcher's creative interpretation.

At times, it is beneficial to limit our search to specific areas of knowledge to identify interesting questions.

Definition 15.3.3 Let $\mathcal{A} \subset \mathcal{T}$ be a research area, and $t_1, t_2 \in \mathcal{T}$ be two topics. If both topics belong to the same area, meaning $t_1, t_2 \in \mathcal{A}$, we say that the question $Q_{t_1 \rightarrow t_2}$ is *intradisciplinary*, otherwise, we say that the question is *interdisciplinary*.

The most innovative questions tend to be interdisciplinary, as they have a lower likelihood of having been considered previously. This is because they require collaboration between specialists from different research areas. Interdisciplinary questions address our requirement **C1** for interesting questions, which dictates that the question must be new and original.

15.4 New Research Topics

TODO: Review this section

In the previous section our focus was in how to find new interesting research questions. In this section we will go one step beyond, and we will show how to identify new, previously unknown, research topics.

Definition 15.4.1 In the two-dimensional space defined by relevance and nescience, the *unknown frontier*, denoted by \mathbb{F} , is defined as the following arc:

$$\mathbb{F} = \{(x, y) \mid x^2 + y^2 = \max(\{N_t^2 + R_t^2, t \in T'\}), x > 0, y > 0\}$$

If we plot all the known research topics according to their relevance and nescience, the unknown frontier will cover them. Intuitively the *unknown frontier* marks the frontier between what we do not know and we are aware that we do not know (we do not fully understand those topics), and what we do not know and we are not yet aware that we do not know those topics. This intuitive property is in general terms, since it may happen that some unknown topics lie under the unknown frontier as well.

Definition 15.4.2 Let T' the set of known research topics. The *new topics area*, denoted by \mathbb{S} , is defined by:

$$\mathbb{S} = \{(x, y) \mid x^2 + y^2 > \max(\{N_t^2 + R_t^2, t \in T'\}), x > 0, y > 0\}$$

The new topics area contains all those unknown topics that we are not aware we do not know them (unknown unknown). The big issue is how to reach this new topics area if we do not know anything about the topics included in that area.

Proposition 15.4.1 Let $r \in T$ be a topic, if $N_r^2 + R_r^2 > \max(\{N_t^2 + R_t^2, t \in T'\})$ then $t \in T \setminus T'$.

Proof. Let $N_r^2 + R_r^2 > \max(\{N_t^2 + R_t^2, t \in T'\})$ and suppose that $r \in T'$, then have that $\max(\{N_t^2 + R_t^2, t \in T'\}) \geq N_r^2 + R_r^2$ that it is a contradiction, and so $t \in T \setminus T'$. ■

Proposition 15.4.1 together with the fact that the combined nescience of two topics is higher than the nescience of any of them isolated (Proposition XXX), and that their combined relevance is higher than the relevance of any of them (Proposition 15.1.1), a possible approach to identify new topics could be by means of combining already existing interesting problems. In Figure 15.2 is depicted graphically the idea.

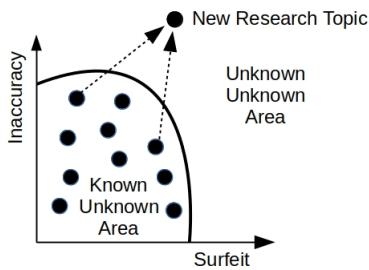


Figure 15.2: The discovery of new research topics

Definition 15.4.3 Given two topics $t_1, t_2 \in T'$, a *new topic*, denoted by $S_{\{t_1, t_2\}}$, is the unordered pair $\{t_1, t_2\}$.

The exact meaning of the new topic that results as the combination of topics t_1 and t_2 is left to the creative interpretation of the researcher.

Definition 15.4.4 The *interestingness* of the new topic, denoted by $IS_{\{t_1, t_2\}}$, is given by:

$$IS_{\{t_1, t_2\}} = R_{t_1}R_{t_2} + N_{t_1}N_{t_2}$$

In practice, what we have to do is to compute all possible combination of those topics with very large interestingness as problems IP_i with themselves, and select the combinations with higher IS .

Of course, some of the combinations generated would be totally meaningless. Advanced techniques from the area of natural language processing or machine learning could be used to try filter out those nonsense combinations.

Definition 15.4.5 Let $A \subset T$ a research area, and $t_1, t_2 \in T$ two topics. If both topics belongs to the same area, that is $t_1, t_2 \in A$, we say that the new topic $S_{\{t_1, t_2\}}$ is *intradisciplinary*, otherwise, we say that the topic is *interdisciplinary*.

Again, the most innovative new topics would be by the combination of interdisciplinary topics, because the probability that somebody has already though about them is lower.

15.5 Classification of Research Areas

In the same way we studied the nescience of research areas (see Section 14.5), we could also study the interestingness of research areas.

Definition 15.5.1 Given a research area $A \subset T$, we define the *average interestingness of the area as a source of interesting tools* by

$$IT_A = \frac{1}{n} \sum_{t \in A} IT_t$$

and the *average interestingness of the area as a source of interesting problems* by

$$IP_A = \frac{1}{n} \sum_{t \in A} IP_t$$

where n is the cardinality of A .

In this way we could compute the interestingness of mathematics, physics, biology, social sciences, and other disciplines as a source of interesting tools and problems. Other alternative measures of centrality and dispersion could be used for the characterization of research areas as well.

As I will show in Chapter ??, the interestingness of mathematics as a source of tools is higher than the interest of social sciences, since mathematics is composed of topics with a high applicability that are very well understood, and that is not the case, in general, for social sciences. On the other hand, the interestingness of social sciences as a source of problems is higher than the interest of mathematics, since the topics studied by the social sciences are more relevant to humankind¹ and, in general, not very well understood.

■ **Example 15.5** We could use the interestingness of an area to identify research areas in decay. A knowledge area is in decay (from the research point of view) if it has no enough interesting research problems. For example, although the aerodynamics of zeppelins is not fully understood (still some nescience), it is not longer useful (low relevance), since people does not use zeppelins to travel anymore, and so, the average interestingness is very low. Another example of area in decay is classical geometry: although it is relevant, our understanding of this subject is nearly perfect, since there are almost no unsolved problems, and so, its average nescience is very low. However, on the contrary to what happens in case of the aerodynamics of zeppelins, classical geometry is still very interesting as a source of tools. ■

It is worth to mention that we could add other metrics to provide a finer, or even alternative, characterization of the unknown unknown area. For example, we could add to nescience and

¹Please mind that I am not saying that the topics addressed by mathematics are not relevant to humankind, what I am saying is that, in relative terms, the problems addressed by social sciences have a higher relevance.

relevance a third dimension with the probability that a topic description is true. However, these extended or alternative characterizations will be not considered in this book. Fortunately, the idea of how to reach the unknown unknown area is the same, regardless of the number and the metrics used (as long as these metrics satisfy some minimal mathematical properties, described in Chapters 14 and 15).

References

Popper and falsificationism

Review the following references

- Freitas, A. A. (1999). On Objective Measures of Rule Surprisingness. In Proceedings of the Second European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD '98), pp. 1-9.
- Gureckis, T. M., & Goldstone, R. L. (2009). How You Named Your Child: Understanding The Relationship Between Individual Decision Making and Collective Outcomes. *Topics in Cognitive Science*, 1(4), 651-674.
- Klein, J. T. (1990). Interdisciplinarity: History, Theory, and Practice. Wayne State University Press.
- Kuhn, T. S. (1962). The Structure of Scientific Revolutions. University of Chicago Press.
- Newell, A., & Simon, H. A. (1972). Human Problem Solving. Prentice-Hall.
- Page, S. E. (2007). The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies. Princeton University Press.
- Rescher, N. (1986). The Riddle of Existence: An Essay in Idealistic Metaphysics. University Press of America.
- Schelling, T. C. (1978). Micromotives and Macrobbehavior. W. W. Norton & Company.



16. Advanced Properties

Invert, always invert.
Carl Gustav Jacob Jacobi

This chapter covers more advanced mathematical properties of the concept of nescience. This chapter can be safely skipped by those readers interested only in the applications of nescience. However, it is highly recommended to read it, since it provides a deeper understanding of what nescience is exactly.

TODO: Explain the abstract nature of the axioms vs. the interpretation we have provided in the previous chapters. Mention that perhaps there exists other interpretations.

Since the time of David Hilbert, mathematics is about the study of the properties of abstract objects and their relations, without paying too much attention to what these objects are or represent. Mathematicians create (or discover) abstract frameworks of logic that can have multiple interpretations. It is up to applied scientists to provide those interpretations. In the same way, we can provide an abstract definition of the concept of nescience, and study its properties, without making any explicit reference to science nor to the scientific method. In doing so, we loose the interpretability of our theory, but, on the other side, we could apply the same results to other disciplines.

Extend this introduction with a very short review of the topics covered in the chapter.

16.1 The Axioms of Science

TODO: Say something intelligent here.

In this section we are going to propose a collection of axioms that formalize what is this thing called science.

In the previous section we have provided a formalization of the theory of nescience based in set theory and first order logic. Set theory is not the only available framework in which mathematics can be formalized. We could have used type theory or category theory instead. Having meaningful axioms of our theory in multiple formalization frameworks, such as set theory, type theory, and category theory, will constitute strong evidence that nescience is a fundamental concept in mathematics, and not a mere artifact of a particular formalization.

16.1.1 Model Theory

Our first approximation to the problem of how to formalize the theory of nescience is a collection of axioms based on first-order logic and the ZFC axioms of set theory with equality (see Appendix B). In this section we will also prove some basic results and explain how these axioms relate to the new ideas we have introduced in this book. For this axiomatization we are not going to follow the approach described in the previous chapters, that is, we are not going to explicitly define the quantities of miscoding, surfeit and inaccuracy. Instead, we will introduce the concatenation and conditional operations, list their fundamental properties, and explain how nescience is affected by them. The remaining concepts follow naturally from these basic axioms.

Definition 16.1.1 Let \mathcal{L} be a non-empty set called *language* whose elements are called *strings*.

We define *science* as the first-order logic structure $(\mathcal{L} \mid \lambda, \mathcal{O}, <_N, \oplus, |)$ where:

- (i) $\lambda \in \mathcal{L}$ is distinct element called *neutral*,
- (ii) \mathcal{O} is a relation called *oracle*,
- (iii) $<_N$ is a relation called *nescience*,
- (iv) \oplus is a binary function called *concatenation*, and
- (v) $|$ is a binary function called *conditional*

that satisfies the following axioms:

$$A1 \quad \forall t \in \mathcal{L} \ t\mathcal{O}t.$$

$$A2 \quad \forall s, t \in \mathcal{L} \text{ if } s\mathcal{O}t \text{ then } t\mathcal{O}s.$$

$$A3 \quad \forall r, s, t \in \mathcal{L} \text{ if } r\mathcal{O}s \text{ and } s\mathcal{O}t \text{ then } r\mathcal{O}t.$$

$$A4 \quad \forall t \in \mathcal{L} \ \neg t <_N t.$$

$$A5 \quad \forall s, t \in \mathcal{L} \text{ if } s <_N t \text{ then } \neg t <_N s.$$

$$A6 \quad \forall r, s, t \in \mathcal{L} \text{ if } r <_N s \text{ and } s <_N t \text{ then } r <_N t.$$

$$A8 \quad \forall s, t \in \mathcal{L} \ \oplus(s, t) = \oplus(t, s).$$

$$A9 \quad \forall s \in \mathcal{L} \ \oplus(s, \lambda) = \oplus(\lambda, s) = s.$$

$$A10 \quad \forall r, s, t \in \mathcal{L} \ \oplus(\oplus(r, s), t) = \oplus(r, \oplus(s, t)).$$

$$A11 \quad \forall s \in \mathcal{L} \ |(s, \lambda) = s.$$

$$A12 \quad \forall s \in \mathcal{L} \ |(s, s) = \lambda.$$

Axioms A1, A2 and A3 state that the oracle \mathcal{O} is an equivalence relation; the nescience relation $<_N$ described by Axioms A4, A5 and A6 is a strict partial order; the concatenation operation \oplus defined by Axioms A8, A9 and A10 together with \mathcal{L} forms a free monoid; and Axioms A11 and A12 define the behaviour of the conditional operator $|$.

We still need two more axioms to fully characterize the behaviour of our logic structure. But first we have to introduce some additional concept.

Definition 16.1.2 Let \mathcal{L}/\mathcal{O} be the quotation set defined by the oracle relation over the language set. We call *entity*, denoted by $[e]$, to every class of this quotation set, that is, $[e] \in \mathcal{L}/\mathcal{O}$.

With the next Axiom we require that every entity contains at least one minimal string with respect to the nescience ordering.

Definition 16.1.3 — Axiom A13. For every entity $[e] \in \mathcal{L}/\mathcal{O}$ there exists at least one $r \in [e]$ such that it is minimal with respect to $<_N$, that is, it does not exist an $s \in [e]$ such that $s <_N r$.

Our last axiom refers to how nescience decreases.

Definition 16.1.4 — Axiom A14. Let $t \in \mathcal{L}$ be a non-minimal element, then:

- (i) there exists s such that $\oplus(t, s) <_N t$, or
- (ii) there exists s such that $|(t, s) <_N t$.

Interpretation of the Axioms

The first thing to note about our proposal is that the set \mathcal{E} of entities in which we are interested is not part of the axioms. Instead, the entities are studied indirectly through a language set \mathcal{L} . Furthermore, we have not provided any indication about the nature of the elements of \mathcal{L} besides that it must be a non-empty set. In this sense, \mathcal{L} could be anything that satisfies our axioms. For example, \mathcal{L} could be the set \mathcal{B}^* of finite binary strings, in which case nescience will be based on string length; or it could be the set \mathbb{N} of natural numbers, and derive nescience from the ordering of these numbers¹. This approach of studying the properties of a set without specifying the nature of its elements (the standard approach used in the mathematics of the last century) has some advantages and disadvantages. The main advantage is that perhaps we could apply our theory to other areas (yet to be discovered) for which the theory is not intended. The limitation is that this abstract nature of the axioms makes more difficult to apply the theory in practice, for example, to derive practical algorithms that compute the new proposed metrics.

The only tool we have at our disposal to match the strings of \mathcal{L} with the entities of \mathcal{E} is the oracle, and this matching has to be done in an indirect way. The oracle defines an equivalence relation that splits the set \mathcal{L} into equivalence classes. Each equivalence class corresponds to an entity of \mathcal{E} , and it might happen that not all entities of \mathcal{E} have an associated class (what we have called the unknowable unknown). All the strings are related to some entity, and some of them would be better than others (i.e. lower nescience). No string can be part of two different entities. For each collection of \mathcal{E} there could be more than one valid oracle. The details (inner workings) of how the oracles match strings to entities is not covered by the axioms.

We cannot provide a quantitative measure for the concept of nescience, since numbers are not part of our axioms. Even if we use as underline set \mathcal{L} the set of natural numbers \mathbb{N} , the symbol $+$ and its properties are not part of our logic structure. Instead what we have provided is a relative ordering of the different elements of \mathcal{L} , intuitively, according to how much we do not know given those strings. The ordering has to be partial, i.e. not every pair of elements of \mathcal{L} can be compared. This partial order is applicable not only in case of strings that belong to different entities, but also it can happen with strings that belong to the same entity. The order has also to be strict, i.e. nescience equality is not defined. The problem of assuming a non-strict total order for nescience is that if $s <_N t$ and $t <_N s$ then it should be the case that " $s = t$ ", that is, if two strings have the same nescience, they should be the same string, and this is not necessarily the case. How nescience is assigned to the elements of \mathcal{L} is not covered by the axioms.

The problem of having multiple styles for the same entity is not explicitly addressed by the axioms, although it is implicit through the use of minimal elements (each minimal element could be related to each particular style). We do require that each equivalence class has at least one minimal element. An equivalence class could have more than one minimal element, and we do not require that classes have a minimum. Intuitively, each minima would correspond to each of the valid ways of representing the entity. We do not expect strings from different representation styles to be directly comparable. Finally, we do not require the existence of global minimum nor global minimas for the set \mathcal{L} .

If a string s is not a local minima for its class with respect to the nescience ordering, there must exist another string t with smaller nescience. There are two ways in which we can find this string, that is, to decrease our nescience about the string: either by concatenating the string with another

¹TODO: Be a little bit more specific and mention in each case which one is the neutral element, and how concatenation and conditional could be defined. Ideally, provide a third, non-trivial, example.

one, or by conditioning the string to another one.

If the string s is missing some critical information required to encode the entity in which we are interested, we could extend s with additional symbols, by means concatenating s with other strings with the relevant symbols. If the string s contains non-relevant or even wrong information, we could get rid of that information by assuming that s is the concatenation of two or more strings, and removing the non-relevant substrings. Concatenation also allows us to discover new entities: concatenating two strings that belong to different entities might lead to a new, previously unknown, entity. In this sense, concatenation would implement the exploration part in this schema of exploration/exploitation in which science is characterized.

Conditioning is the tool that we have at our disposal to leverage on already existing knowledge. The difference between concatenation and conditioning is that concatenation adds something to our string, meanwhile conditioning allows us to use something (if needed) without extending our current string. Conditioning would implement the exploitation part in this science schema of exploration/exploitation.

Basic Properties

First of all we will introduce some notational conventions to simplify the algebraic operations dealing with nescience.

Notation 16.1. *We will use the infix notation for the concatenation function, that is, we will write $s \oplus t$ instead of the $\oplus(s, t)$. Moreover, given Axiom A10, we will drop parenthesis in case of multiple concatenations.*

We will use the infix notation for the conditional function, that is, we will write $s | t$ instead of the $|(s, t)$.

It is also convenient to introduce the symbol $=_N$ to represent the case that our unknown given the strings s and t is the same. We will apply this symbol only if both strings belong to the same equivalence class.

Notation 16.2. *Let $[e]$ be an entity, and $s, t \in [e]$ two strings. We denote by $s =_N t$ the case that $(s, t) \notin <_N$.*

Conditional is non-decreasing wrt nescience

Proposition 16.1.1 Let s and t two arbitrary strings, then we have that $s <_N (s | t)$.

Proof. TO DO ■

TODO: Introduce the concepts of minimal and valid.

TODO: Prove that minimal implies valid.

TODO: Prove that joining two valid strings is a valid string.

TODO: Distributive Law - $((r \text{ lc } s) \text{ <+>} t) \text{ <n } ((r \text{ <+>} t) \text{ lc } (s \text{ <+>} t))$.

Axiomatic Definition of the Elements of Nescience

As we have said at the introduction of this section, the collection of proposed axioms do not explicitly mention the concepts of miscoding, inaccuracy and surfeit. Moreover, in the set \mathcal{L} we do not distinguish between representations and models. All these elements are introduced as a particular interpretation of the axioms, the one we have been developing in Part II of this book. In this section we are going to prove that our interpretation of science satisfies the science axioms.

We start by assuming that the underline set \mathcal{L} is the set of all finite binary strings \mathcal{B}^* . We also assume that this set is composed by two kind of strings: representations, that are regular strings, and models, that are strings that encode a particular Turing machine assuming a fixed universal oracle machine. Each equivalence class will contain all possible representations and

descriptions of that entity, including the wrong ones. We have also to assume that the oracle knows if a particular string is a representation or a description, and proceeds accordingly. For example, if the string is a representation, the oracle could do nothing, leaving the string as is, but if the string is a model, then the oracle could run the model (using our universal Turing machine) and use the output as a representation. Distinguishing between representations and descriptions also implies that the operation of concatenation has more sense in case of representations, and the operation of conditioning would be intended for descriptions. Concatenation would be used to include in our representations all the information needed to encode the entity (string concatenation), and conditional would be used to make descriptions shorter, by leveraging on other, already existing, models (conditional complexity).

TODO: Prove that our mododified universal oracle machine is an equivalence relation

TODO: Introduce the concepts os miscoding, surfeit and inaccuracy, and nescience.

TODO: Prove that nescience is a strinct partial order.

TODO: Prove that string concatenation is a free monoid. For the case of regular strings, for the case of models, and for mixed concatenations.

TODO: Prove that the axioms related to conditional satisfied the axioms. For the case of regular strings, for the case of models, and for mixed concatenations.

TODO: Prove that each equivalence class has at least one minimal element.

TODO: Prove that the nescience reduction axiom in fact is satisfied.

16.1.2 Type Theory

In this section we are going to provide a formalization of the theory of nescience in the context of type theory (see Section B.4). Type theory has some advantages over set theory. The most important is that it is a constructive theory, which means that we do not assume the existence of abstract mathematical entities that satisfy some properties, but we show how to construct those entities. For example, there is no need to resort to an abstract, uncomputable, oracle. In type theory, all propositions and proven theorems have an equivalent computable function or algorithm, and thus, we can provide computer programs to solve all problems addressed by our theory (see Section 17.1 for a practical implementation of some of these algorithms in the form of a software library).

A second advantage of type theory is that the lambda terms used in the theory are by definition computable functions, and that lambda calculus is a Turing machine capable of universal computation. In the theory of nesciece we require that our models be lambda terms, and that our universal Turing machine be the lambda calculus. In this sense, our models are native elements within the theory, not external artifacts based on an arbitrary, axiomatically defined, universal machine. The models that describe entities are lambda terms executed in the univeral Turing machine of lambda calculus.

The third advantage is that there are software implementations of type theory, such as the Coq proof assistant (see Section D), which allow us to mechanically verify the correctness of our theory. In the rest of this section, we will use the coq syntax for definitions, propositions and theorems so that interested readers can use a computer and a Coq interpreter to verify by themselves that the proofs are correct. We have also included an appendix where we briefly describe the Coq language, so that those readers who prefer traditional mathematics can translate the Coq scripts into classical mathematical definitions and propositions.

Foundamental Concepts

We start by introducing a new type, called `String`. The type string is defined recursively in the following way: the empty string, called `lambda`², is a string, and the sucessor of a string, given the function `S`, is also a string. For us strings are just a list of lambdas, and our theory is based on

²Do not confuse the λ symbol with a λ -term.

the collection of all finite strings $\{\lambda, \lambda\lambda, \lambda\lambda\lambda, \dots\}$. Both descriptions and representations will be finite strings of lambdas.

```
Inductive String : Set :=
| lambda : String
| S       : String -> String.
```

Before to introduce the main concepts of the theory of nescience, we have to define a helper function called `Difference`. The `Difference` function computes the absolute difference between two strings, that is, the excess of lambdas of the longer string with respect to the shortest one. For example, if we have the strings $\lambda\lambda$ and $\lambda\lambda\lambda$, the difference would be the string λ . Given two strings r and s , `Difference` is defined recursively as:

```
Fixpoint Difference (r s : String) : String :=
match r, s with
| lambda, lambda => lambda
| lambda, s        => s
| r    , lambda   => r
| S r' , S s'    => Difference r' s'
end.
```

We say that a string r has smaller *surfeit* than a string s if the string r is shorter, in terms of number of lambdas, than the string s . Given two strings r and s , the `Surfeit` function is defined recursively, and returns `true` if the string r has less lambdas than the string s . Applied to descriptions, we prefer those with the smaller number of lamdas.

```
Fixpoint Surfeit (r s : String) : bool :=
match r, s with
| lambda, lambda => false
| lambda, _        => true
| _, lambda      => false
| S r' , S s'    => Surfeit r' s'
end.
```

The *inaccuracy* of a string r with respect to a target string t is based on the absolute difference between the two strings, that is, how close is the string to the target. Given two strings r and s , and a target string t , the `Inaccuracy` function returns `true` if the absolute difference between the strings r and t contains less lambdas than the absolute difference between the strings s and t . Inaccuracy will be applied to the ouput of the models.

```
Definition Inaccuracy (r s : String) (t : String) : bool :=
Surfeit (Difference r t) (Difference s t).
```

Nescience will be based on the concepts of surfeit and inaccuracy. Intitively, we are looking for very short descriptions (low surfeit) and with very low error (low inaccuracy). The `Nescience` function gets as input five strings: mr, ms, r, s and e , and returns `true` if mr is a shorter string than ms , and the inaccuracy of r is smaller than the inaccucay of s with respect to e . Intuitively, mr and ms would be a string-based representations of our models (in our case, lambda-terms), r and s are the string-based outputs of these models, and e is a string-based representation of an entity.

```
Definition Nescience (mr ms : String) (r s : String) (e : String) : bool :=
andb (Surfeit mr ms) (Inaccuracy r s e).
```

In practice, we generally do not know a string-based representation e of the entity we are interested in, so an indirect approach must be used to study that entity. *Miscoding* is this approach, and it will be introduced in a later section.

Instead of defining the new type `String` we could have reused the concept of natural number, that is, we could have introduced descriptions and representations as numbers. In that case, the `Difference` between two strings r and s would be the absolute difference $|r - s|$ between the

numbers r and s , and the Surfeit could be based on the strict order relation between numbers $<$. The concepts of Inaccuracy and Nescience would have been defined in the same way with numbers. However, natural numbers require additional properties that are not needed for the theory of nescience. For example, the multiplication of two natural numbers does not make any sense in our theory. Reusing natural numbers would have made our theory unnecessarily complex.

In the rest of this section, we will see how we can derive our theory of nescience using only these fundamental concepts. We will also prove the most significant results and derive algorithms for applying the theory in practice.

Surfeit

Surfeit allow us to compare two strings. Recall that the surfeit of two strings r and s is true if, and only if, s is composed by more lambdas than r . Surfeit allow us to order the models (string based representations of the models) by its length. We are interested in those models with the lowest possible complexity. In the rest of this section we are going to review the properties of surfeit.

The surfeit of a string does not increases when we conditions that string to another one. Intuitively, assuming some previous background knowledge already known as true allow us to reduce the complexity of our models.

```
Proposition SurfeitConditional : forall r s : String,
  Surfeit r (r |c s) = false.
Proof.
intros.
induction r as [| r' IHr'].
- simpl.
  reflexivity.
-
Admitted.
```

From a theoretical point of view, the limit to the process of conditioning would be to assume as true the current model.

```
Proposition SurfeitConditionalLambda : forall r s: String,
  Surfeit lambda (r |c r) = false.
Proof.
Admitted.
```

The surfeit of a string does not decreases when it is concatenated with another string. Intuitively, the concatenation of two models increases our unknown. This process of concatenating models can be used as an exploratory mechanism to discover new research topics (previously unknown entities).

```
Proposition SurfeitConcatenation : forall r s : String,
  Surfeit r (r <+> s) = false.
Proof.
intros.
Admitted.
```

From the point of view of surfeit, the operation of concatenation can be distributed over the operation of conditioning, as next proposition shows (TODO: provide the intuition behind this distributive law).

```
Proposition DistributiveLawSurfeit : forall r s t : String,
  Surfeit ((r |c s) <+> t) ((r <+> t) |c (s <+> t)) = true.
Proof.
Admitted.
```

Inaccuracy

Definition and results about inaccuracy

Inaccuracy does not increases with conditional

```
Proposition InaccuracyConditional : forall r s e : String,
  Inaccuracy r (r |c s) e = false.
Proof.
Admitted.
```

The limit would be conditioning a string to itself

```
Proposition InaccuracyConditionalLambda : forall r s: String,
  Inaccuracy lambda (r |c r) = false.
Proof.
Admitted.
```

Distributive Law

```
Proposition DistributiveLawInaccuracy : forall r s t : String,
  Inaccuracy ((r |c s) <+> t) ((r <+> t) |c (s <+> t)) = true.
Proof.
Admitted.
```

Entities, Representations and Descriptions

After we have described the fundamental concepts of the theory of nescience, in this section we are going to introduce the remaining elements of the theory, that is, the concepts of entity, representation and description.

We introduce the concept of *entity* recursively as a finite list of strings. Those strings correspond to the valid representations of the entity. We consider that the empty entity, denoted by *nil*, is also an entity.

```
Inductive Entity : Set :=
| nil      : Entity
| add_repr : String -> Entity -> Entity.
```

A *universe* is a finite list of entities. A universe correspond to the particular research area in which we are interested. We consider that the empty universe, denoted by *empty*, is also a universe. The recursive definition of universe is given by:

```
Inductive Universe : Set :=
| empty      : Universe
| add_entity : Entity -> Universe -> Universe.
```

A description is a pair composed by a recursive function (a λ -term) from strings to strings, what we call a *model*, and an input string to that model.

```
Definition Description (model : String -> String) (input : String) : String :=
  model input.
```

Miscoding

Definition and results about miscoding

Properties of Strings

Before we move into the details of the theory of nescience, we are going to prove some basic properties that our concept of string satisfies. Also, besides to the already introduced concept of string difference, we are going to introduce two other strings operators: concatenation and conditional.

We say that two strings are *equal* if, and only if, they have the same number of sigmas. String equality is defined recursively as follows:

```
Fixpoint Equality (r s : String) : bool :=
  match r, s with
  | lambda, lambda => true
  | lambda, _       => false
```

```
| _      , lambda => false
| S r' , S s'  => Equality r' s'
end.
```

It is convenient to introduce a new infix operator to represent the string equality concept. That will help us to simplify new definitions and proofs.

```
Notation "x =? y"  := (Equality x y)
```

In the same way, we introduce another infix operator to represent the concept of string difference.

```
Notation "x <d> y"  := (Difference x y)
```

Strings with the difference operator form an abelian group. That is: it has a neutral element, it is commutative, associative and has an inverse element.

The neutral element for strings is the λ symbol. That λ is the neutral element with respect to the operation of difference is a direct consequence of the definitions of string and difference.

```
Proposition DifferenceNeutral : forall r : String,
  r <d> lambda = r /\ lambda <d> r = r.
Proof.
intros.
split.
- destruct r.
  * reflexivity.
  * reflexivity.
- destruct r.
  * reflexivity.
  * reflexivity.
Qed.
```

The property of being commutative of the difference operator is proved by ... TODO

```
Proposition DifferenceCommutative : forall r s : String,
  r <d> s = s <d> r.
Proof.
Admitted.
```

In the same way, the property of being commutative is shown by ... TODO

```
Proposition DifferenceAssociative : forall r s t : String,
  r <d> (s <d> t) = (r <d> s) <d> t.
Proof.
Admitted.
```

The commutative property can be generalized to finite collections of strings, such that the use of paraenthesis is not needed ... TODO

Given an arbitrary string r , its inverse element is the string itself, as next proposition shows. The proposition is proved by induction over r .

```
Proposition DifferenceInverse : forall r : String,
  r <d> r = lambda.
Proof.
intros.
induction r as [| r' IHr'].
- reflexivity.
- simpl.
  apply IHr'.
Qed.
```

The operation of *concatenation* of two strings returns a new string composed by one of the strings appened at the end of the other. The operation of concatenation is defined recursively as:

```
Fixpoint Concatenate (r s : String) : String :=
  match r, s with
  | lambda, _      => s
  | S r' , _       => S (Concatenate r' s)
  end.
```

The infix notation for the concatenation operation is given by:

```
Notation "x <+> y" := (Concatenate x y)
```

The pair composed by strings and the concatenation operation form a commutative monoid. That is, it has a neutral element, and it satisfies the properties of being commutative and associative.

The neutral element of the operation of concatenation is λ . In order to prove that λ is indeed the neutral element we ... **TODO**

```
Lemma s_plus_lambda : forall s : String,
  s <+> lambda = s.
Proof.
intros.
induction s as [| s' IHs'].
- simpl.
  reflexivity.
- simpl.
  rewrite -> IHs'.
  reflexivity.
Qed.
```

Given the above lemma we can now prove that λ is the neutral element ... **TODO**

```
Proposition ConcatenationNeutral : forall r : String,
  r <+> lambda = r /\ lambda <+> r = r.
Proof.
Admitted.
```

TODO: string concatenation is commutative, and its proof requires to additional lemmas.

```
Lemma plus_r_Ss : forall r s: String,
  S (r <+> s) = r <+> S s.
Proof.
intros.
induction r as [| r' IHr'].
- simpl.
  reflexivity.
- simpl.
  rewrite -> IHr'.
  reflexivity.
Qed.
```

```
Lemma S_equal : forall n m : String,
  n = m -> S n = S m.
Proof.
intros.
induction m as [| m' IHm'].
- rewrite -> H.
  reflexivity.
- rewrite -> H.
  reflexivity.
Qed.
```

```
Proposition ConcatenationCommutative : forall r s : String,
  r <+> s = s <+> r.
Proof.
intros.
induction s as [| s' IHs'].
- simpl.
```

```

apply s_plus_lambda.
- simpl.
rewrite <- IHs'.
rewrite <- plus_r_Ss.
reflexivity.
Qed.
```

Associative property of concatenation ...

```

Proposition ConcatenationAssociative : forall r s t : String,
  r <+> (s <+> t) = (r <+> s) <+> t.

Proof.
intros.
induction r as [| r' IHr'].
- simpl.
  reflexivity.
- simpl.
  rewrite <- IHr'.
  reflexivity.
Qed.
```

Introduce the operator of string conditional, as a recursive definition.

```

Fixpoint Conditional (r s : String) : String :=
  match r, s with
  | lambda, _           => lambda
  | s , lambda          => s
  | S r' , S s'         => Conditional r' s
  end.
```

Infix notation.

```
Notation "x |c y" := (Conditional x y).
```

Algebraic structure of the conditional operator.

Left Neutral Element

```

Proposition ConditionalLeftNeutral : forall r : String,
  r |c lambda = r.

Proof.
intros.
destruct r.
- reflexivity.
- reflexivity.
Qed.
```

Inverse Element

```

Proposition ConditionalInverse : forall r : String,
  r |c r = lambda.

Proof.
intros.
induction r as [| r' IHr'].
- simpl.
  reflexivity.
- simpl.
Admitted.
```

Having introduced the operations of difference, concatenation and conditional, we can study which of these operators can be distributed over the others, since not all possible combinations are valid. In fact, only three distributive laws are true in our theory of nescience.

The operation of concatenation can be distributed to the operation of conditional, as the next proposition shows. The proposition is proved by ... TODO.

```

Proposition DistributiveConditionalConcatenation : forall r s t : String,
  Surfeit ((r |c s) <+> t) ((r <+> t) |c (s <+> t)) = true.
Proof.
Admitted.

```

The concatenation operator can also be distributed to the difference operator. The proposition is proved by ... TODO.

```

Proposition DistributiveDifferenceConcatenation : forall r s t : String,
  Surfeit ((r <d> s) <+> t) ((r <+> t) <d> (s <+> t)) = true.
Proof.
Admitted.

```

Finally, we have a third distributive law, that says that the difference operator can be distributed to the concatenation. Th proposition is proved by ... TODO.

```

Proposition DistributiveConcatenationDifference : forall r s t : String,
  Surfeit ((r <+> s) <d> t) ((r <d> t) <+> (s <d> t)) = true.
Proof.
Admitted.

```

Equivalence of Axioms

In type theory the underline set \mathcal{S} is the collection of all finite unary strings.

Surfeit is based on string length.

Inaccuracy is based on the length of the absolute difference between the string an the target.

Nescience is based on lower surfeti and lower inaccuracy

* strict * antisymmetric * transitive

When studying the properties of strings we have shown that the string concatenation operation is commutative, asociative and has a neutral element, and that string conditional has a left neutral element and an inverse element.

16.1.3 Category Theory

TODO: Interpretation of the theory of nescience in terms of category theory.

16.2 Scientific Method

Next definition formally introduces the concept of scientific methodology in the context of the theory of nescience.

Definition 16.2.1 A *scientific methodology* is an effective procedure that produces a sequence of $t_1, t_2, \dots, t_n, \dots$ where $t_i \in \mathcal{D}$ such that $N(t_i) < N(t_{i+1})$.

Note that we do not require that in a scientific methodology the collection of t_i refer to the same entity.

Describe our proposal of scientific method. Short story: based on a exploration/exploitation approach, where the exploration is based in the concept of joint descriptions, and the exploitation in the concept of conditional description.

Compare against another proposals of scientific method (inductive-deductive, hypothetico-deductive, etc) and with other techniques for knowledge discovery and creativity (triz, etc.)

16.2.1 Science as a Language

TODO: Provide an alternative definition of the set of descriptions, and prove that it is equivalent to our definition based on the description function

Since we require computable descriptions, we would like to know if the set of all possible descriptions of a topic is computable as well.

Proposition 16.2.1 Study if D_t is Turing-decidable, Turing-recognizable or none.

Proof. TODO

Although we know that it is not computable, we are interested in the set composed by the shortest possible description of each topic.

Definition 16.2.2 We define the set of *perfect descriptions*, denoted by \mathcal{D}^* , as:

$$\mathcal{D}^* = \{d_t^* : t \in \mathcal{T}\}$$

Since the set \mathcal{D} includes all the possible descriptions of all the possible (describable) topics, we can see this set as a kind of language for science. We do not call it universal language since it depends on the initial set of entities \mathcal{E} and the particular encoding used for these entities.

Proposition 16.2.2 The set \mathcal{D} is not Turing-decidable.

Proof. TODO: Because it depends on the set \mathcal{T} that it could be not computable

Say something here

Proposition 16.2.3 The set \mathcal{D} is Turing-recognizable.

Proof. TODO: The same argument as the previous proposition

A universal language is determined by a universal Turing machine. Given Two different universal Turing machines δ_a and δ_b defines two different universal languages \mathcal{L}_a and \mathcal{L}_b . Let $\mathcal{L}_{a=b} = \{\langle d_a, d_b \rangle, d_a, d_b \in \mathcal{D} \mid \delta_a(d_a) = \delta_b(d_b)\}$.

Proposition 16.2.4 Study if $\mathcal{L}_{a=b}$ is Turing-decidable, Turing-recognizable or none.

Proof. TODO

Let $\mathcal{L}_{\#t}$ the laguage of valid descriptions (from a formal point of view) that do not describe any topic, that is, $\mathcal{L}_{\#t} = \{d \in \mathcal{D} \mid \nexists t \in \mathcal{T}, \delta(d) = t\}$.

Proposition 16.2.5 Study if $\mathcal{L}_{\#t}$ is Turing-decidable, Turing-recognizable or none.

Let $\mathcal{L}_{\#d}$ the laguage topics that are not described by any description, that is, $\mathcal{L}_{\#d} = \{t \in \mathcal{T} \mid \nexists d \in \mathcal{D}, \delta(d) = t\}$.

Proposition 16.2.6 Study if $\mathcal{L}_{\#d}$ is Turing-decidable, Turing-recognizable or none.

TODO: Extend the concept of conditional description to multiple topics.

TODO: Introduce the concept of "independent topics" based on the complexity of the conditional description. Study its properties.

TODO: Define a topology in \mathcal{T} and \mathcal{D} . Study the continuity of δ . Study the invariants under δ .

16.3 The Inaccuracy - Surfeit Trade-off

TODO: Explain that given a miscoding, there is a trade-off between inaccuracy and surfeit, in the sense that there exists an optimal point beyond it the more we decrease the inaccuracy, the more we increase the surfeit, and so, the nescience keep constant. Explain how this trade-off imposes a limit to knowledge.

16.4 Science vs. Pseudoscience

TODO: Provide a characterization of the difference between science and pseudoscience. In science, nescience decreases with time, in pseudoscience not.

16.5 Graspness

There are some topics whose nescience decrease with time much faster than others, even if the amount of research effort involved is similar. A possible explanation to this fact is that there are some topics that are inherently more difficult to understand.

We define the *graspness* of a topic as the entropy of the set of possible descriptions of that topic. A high graspness means a difficult to understand topic. Intuitively, a research topic is difficult if it has no descriptions much shorter than the others, that is, descriptions that significantly decrease the nescience. For example, in physics there have been a succession of theories that produced huge advances in our understanding of how nature works (Aristotelian physics, Newton physics, Einstein physics), meanwhile in case of philosophy of science, new theories (deductivism, inductivism, empiricism, falsation, ...) have a similar nescience than previous ones.

Definition 16.5.1 — Graspness. Let D_t the set of descriptions of a topic $t \in T$. The *graspness* of topic t , denoted by $G(t)$, is defined by:

$$G(t) = \sum_{d \in D_t} 2^{-l(d)} l(d)$$

Graspness is a positive quantity, whose maximum is reached when all the descriptions of the topic have the same length:

Proposition 16.5.1 Let $D_t = \{d_1, d_2, \dots, d_q\}$ the set of descriptions of a topic $t \in T$, then we have that $G(t) \leq \log q$, and $G(t) = \log q$ if, and only if, $l(d_1) = l(d_2) = \dots = l(d_q)$.

Proof. Replace $P(s_i)$ by $2^{-l(d_i)}$ in Proposition 5.4.1. ■

If $G(t) = \log q$ then we have that $N_t = 0$ for all \hat{d}_t . In that case, it does not make any sense to do research, since no new knowledge can be acquired. This is the case of pseudosciences, like astrology, where the nescience of descriptions does not decrease with time. If fact, graspness can be used as a distinctive element of what constitutes *science* and what does not.

Definition 16.5.2 A topic $t \in T$ is *scientable* if $G(t) < \log d(D_t) - \varepsilon$, where $\varepsilon \in \mathbb{R}$ is a constant.

We can extend the concept of graspness from topics to areas, for example, by means of computing the average graspness of all the topics included in the area.

Definition 16.5.3 Let $A \subset T$ a research area. The *graspness* of area A , denoted by $G(A)$, is defined by:

$$G(A) = \frac{1}{d(A)} \sum_{t \in A} G(t)$$

16.6 Effort

TODO: Provide a characterization of the effort, measured in terms of number of operations, or time, to reduce the nescience. Based in the concept of computational complexity. Provide a physical interpretation in terms of information.

16.7 Human Understanding

In his landmark paper "*On Computable Numbers with an Application to the Entscheidungsproblem*", where the concept of computable function was proposed for the first time, when the author Alan M. Turing talked about computers, he was thinking about human computers, not machines. In this

sense, according to Turing, everything that is computable, can be computed by a human, at least in theory.

The first limitation is about computation time. We expect that a human is able to find a solution to a particular instance of a problem in order of seconds, maybe minutes or hours, but definitely, in less than a life time, otherwise another human have to start again from scratch to solve the problem. So, given the average computing speed of a human brain, we identify as human solvable problems only those problems with a complexity smaller than a fixed number of steps.

The second limitation is about program size. It might happen that the algorithms to solve a problem is simply too big to fit in our brains. Of course, we could argue that as we saw in Example XX, only X states and Y tape symbols are sufficient to implement a universal Turing machine capable of solving any computable problem, that is, any problem for which exist a Turing machine that can solve it. However, by just following a set of instruction we do not mean that we understand a problem. We understand a problem when we have made the problem for ourselves, in the sense, that the problem is somehow stored in our brain in our own language. So this limits the problem to those whose length, including the necessary background, can be stored in our brain.

We are looking for solutions that minimize at the same time the computational complexity and the Kolmogorov complexity.

Definition 16.7.1 We say that a problem P is *human solvable* if there exist an algorithm TM to solve P such that $t(n)$ and $K(P) < l$.

Given the above definition, we could argue that most of the problems we know are not human solvable, but in fact, they have been solved by human. We use two strategies to deal with too complex problems for our individual brains. The first strategy is that those time-consuming mechanical parts of the problems are left for computers, so we can reduce the computing time. The second one is that we split the problems into subproblems and each of use specializes in those subproblems.

In this section we are interested in to study the nature of these problems in which this strategy cannot be applied, and so, they are out of reach of individual, nor groups of, humans.

16.8 Areas in Decay

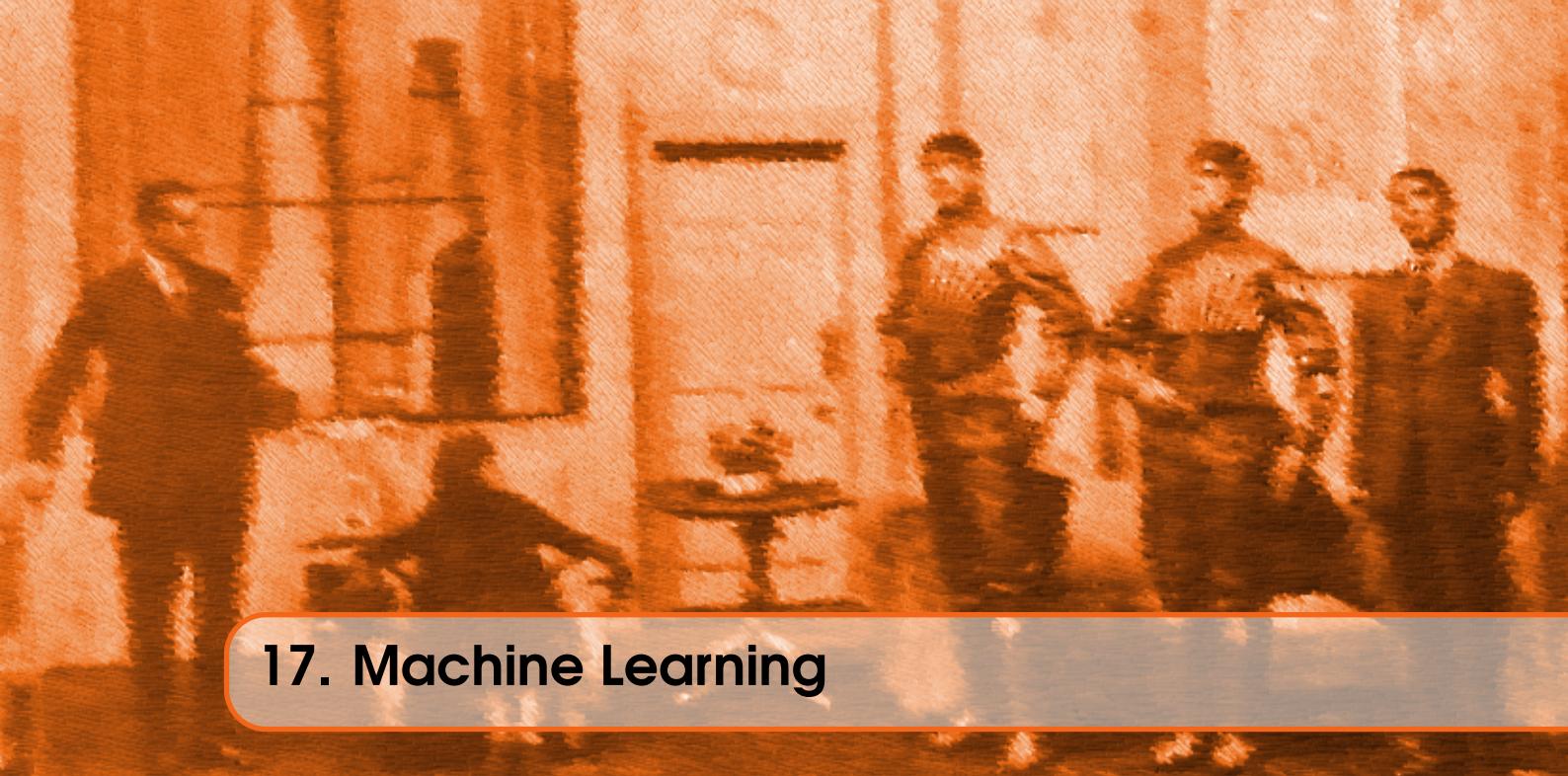
Provide a model of the decay in the interest of research areas. For example, a good explanatory variable could be the number of interesting questions: the less interesting questions, the more the area is near to its end as a interesting research area, of course, as long as its topics are not used as tools.

References

Mention the polemic between Hilbert and Fregde given a reference to the book of Mosterin.

Part 3: Applications

17	Machine Learning	237
17.1	Nescience Python Library	
17.2	A Note About Compression	
17.3	Miscoding	
17.4	Inaccuracy	
17.5	Surfeit	
17.6	Nescience	
17.7	Auto Machine Classification	
17.8	Auto Machine Regression	
17.9	Time Series	
17.10	Anomaly Detection	
17.11	Decision Trees	
17.12	Algebraic Model Selection	
17.13	The Analysis of the Incompressible	
18	Software Engineering	277
18.1	Redundancy of Software	
18.2	Quality Assurance	
18.3	Forex Trading Robots	
19	Philosophy of Science	285
19.1	Wikipedia, The Free Encyclopedia	
19.2	Classification of Research Topics	
19.3	Classification of Research Areas	
19.4	Interesting Research Questions	
19.5	Interesting Research Topics	
19.6	Philosophy of Science	
19.7	Evolution of Knowledge	
19.8	Graspness of Topics	
19.9	Probability of Being True	
19.10	Unused text	
20	Computational Creativity	301
20.1	Classification of Research Topics	
20.2	Interesting Research Questions	
20.3	New Research Topics	
20.4	Classification of Research Areas	
20.5	References	
20.6	Future Work	



17. Machine Learning

*There are no difficult problems,
only lack of imagination.*

Antonio García

We have seen that the most difficult problems to which we can apply the results of the theory of nescience arise when set of entities \mathcal{E} under study is composed by abstract elements. The difficulty with abstract entities is that it does not exists a way to encode them as strings of symbols so we can effectively reconstruct them. In practice, a possible approach to deal with this problem is to run an experiment and collect the results, as we do in case of physics. An alternative approach would be to take a collection of measurements, like for example, by means of observing the behavior of users in an online social network.

This chapter is devoted to how to apply the concept of minimum nescience to the area of machine learning. We assume that the entities under study are encoded as a dataset \mathbb{X} composed by n training vectors of p predictors and a response variable \mathbf{y} (see Section 7.2).

We will start by providing practical approximations for the concepts of miscoding, inaccuracy and surfeit when the entities are encoded as datasets, and then we will show how to combine them in the single quantity of nescience. These approximations will allow us to introduce the *minimum nescience principle*, a technique designed to automate the process of finding optimal models in machine learning (auto-machine learning).

Besides introducing these approximations, we will show how to apply them to solve practical problems. The examples will be based on the `nescience`¹ library, an open source python library that provides an implementation of the ideas included in this chapter.

17.1 Nescience Python Library

The `nescience` library is an open source Python library that provides an implementation of the ideas included in this book applied to the area of machine learning. The library follows the API

¹<https://github.com/rleiva/nescience>

and conventions of the highly popular `scikit-learn` machine learning tool suite, and so, it can be combined with the methods provided by this package.

The `nescience` library can be installed with the `pip` utility:

```
pip install nescience
```

In the web page that accompanies this book, the reader can find a collection of notebooks for the `jupyter-lab` environment describing how the library works. For each subsection of this chapter, there is a notebook that implements all the examples included, so that the reader can repeat and play with them. Additional information about the `nescience` library, and a reference of the API provided, can be also found in the web page of the book.

17.2 A Note About Compression

As it is customary, the Kolmogorov complexity $K(s)$ of a string s will be approximated by the length of the compressed version of that string using a standard compressor, that is, we will use the normalized compression distance:

$$E_Z(\mathbf{x}_j, \mathbf{y}) = \frac{\max\{\hat{K}_Z(\mathbf{x}_j | \mathbf{y}), \hat{K}_Z(\mathbf{y} | \mathbf{x}_j)\}}{\max\{\hat{K}_Z(\mathbf{x}_j), \hat{K}_Z(\mathbf{y})\}}$$

where $\hat{K}_Z(s)$ denotes the length of the compressed version of the string s using the compressor Z . In the particular case of having a vector $\mathbf{x} = \{x_1, \dots, x_n\}$ of measurements, the string s to be compressed will be the concatenation of the encoded values $s = \langle x_1, \dots, x_n \rangle$. We prefer the following equivalent definition of normalized compression distance, since in practice it is easier to compute the joint distribution of two vectors than the conditional distribution:

$$E_Z(\mathbf{x}_j, \mathbf{y}) = \frac{\hat{K}_Z(\mathbf{x}_j, \mathbf{y}) - \min\{\hat{K}_Z(\mathbf{x}_j), \hat{K}_Z(\mathbf{y})\}}{\max\{\hat{K}_Z(\mathbf{x}_j), \hat{K}_Z(\mathbf{y})\}}$$

As compression technique we will use a code C with minimal length, given the relative frequencies of the different observed values (see Section 5.3). If \mathbf{x} is a qualitative vector (either a feature or the target variable) taking values from a set labels $\mathcal{G} = \{g_1, \dots, g_\ell\}$, that is $\mathbf{x} \in \mathcal{G}^n$, the quantity $\hat{K}_C(\mathbf{x})$ can be computed as:

$$\hat{K}_C(\mathbf{x}) = - \sum_{i=1}^l \log_2 \frac{\sum_{j=1}^n I(x_j = g_i)}{n}$$

If the case of \mathbf{x} being based on a continuous random variable, we cannot calculate the probability of x_j since, in general, the underline probability distribution of \mathbf{x} is unknown. Moreover, we have that $P(x_j) = 0$ for all j . In order to approximate the value $K(\mathbf{x})$ using a minimal length code C , we have to discretize first the vector \mathbf{x} into a collection of intervals.

A discretization algorithm is a mapping between a (possibly huge) number of numeric values and a reduced set of discrete values, and so, it is a process in which some information is potentially lost. The choice of discretization algorithm is something that could have a high impact in the practical computation of the `nescience`. We are interested in a discretization algorithm that produces a large number of intervals (low bias), with a large number of number of observations per interval (low variance). Common techniques include *equal width discretization*, *equal frequency discretization* and *fixed frequency discretization*. However, these techniques require the optimization of an hyperparameter, and so, they are not suitable for our purposes.

In the `nescience` library we use a proportional discretization approach (see Section 5.6), where the number of intervals m and the number of observations per interval s are equally proportional to the number of observations n . In particular, in the `nescience` library we set $s = m = \sqrt{n}$.

Using this discretization procedure, we can approximate the Kolmogorov complexity of a vector \mathbf{x} by:

$$\hat{K}_C(\mathbf{x}) = - \sum_{i=1}^m \log_2 \frac{\sum_{j=1}^n I(x_j \in D_i)}{n}$$

where D_i is the interval defined by the end points $(i-1, i)$.

The quantity \hat{K}_C can be generalized to the an arbitrary number of m vectors $\hat{K}_C(\mathbf{x}_1, \dots, \mathbf{x}_m)$ composed by n samples each, by considering the joint encoded vector

$$\langle \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m \rangle = \{ \langle x_{11}, x_{12}, \dots, x_{1m} \rangle, \dots, \langle x_{n1}, x_{n2}, \dots, x_{nm} \rangle \}$$

Proposition 17.2.1 The normalized compression distance of two vectors x and y computed using a compressor based on optimal codes is equivalent to the normalized mutual information of these two vectors, that is:

$$NCD_C(\mathbf{x}, \mathbf{y}) = 1 - \frac{I(\mathbf{x}; \mathbf{y})}{\max\{H(\mathbf{x}), H(\mathbf{y})\}}$$

Proof. We have to prove that

$$\begin{aligned} NCD(\mathbf{x}, \mathbf{y}) &= \frac{C(\mathbf{x}, \mathbf{y}) - \min\{C(\mathbf{x}), C(\mathbf{y})\}}{\max\{C(\mathbf{x}), C(\mathbf{y})\}} = \frac{nH(\mathbf{x}, \mathbf{y}) - \min\{nH(\mathbf{x}), nH(\mathbf{y})\}}{\max\{nH(\mathbf{x}), nH(\mathbf{y})\}} \\ &= \frac{H(\mathbf{x}, \mathbf{y}) - \min\{H(\mathbf{x}), H(\mathbf{y})\}}{\max\{H(\mathbf{x}), H(\mathbf{y})\}} \end{aligned}$$

We have to consider two cases. In case 1 we assume that $H(\mathbf{x}) > H(\mathbf{y})$, and so

$$\begin{aligned} NCD(\mathbf{x}, \mathbf{y}) &= \frac{H(\mathbf{x}, \mathbf{y}) - H(\mathbf{y})}{H(\mathbf{x})} = \frac{H(\mathbf{y}) + H(\mathbf{x} | \mathbf{y}) - H(\mathbf{y})}{H(\mathbf{x})} = \frac{H(\mathbf{x} | \mathbf{y})}{H(\mathbf{x})} \\ &= \frac{H(\mathbf{x}) - (H(\mathbf{x}) - H(\mathbf{x} | \mathbf{y}))}{H(\mathbf{x})} = 1 - \frac{I(\mathbf{x}; \mathbf{y})}{H(\mathbf{x})} \end{aligned}$$

and in case of $H(\mathbf{y}) > H(\mathbf{x})$ we have that

$$\begin{aligned} NCD(\mathbf{x}, \mathbf{y}) &= \frac{H(\mathbf{x}, \mathbf{y}) - H(\mathbf{x})}{H(\mathbf{y})} = \frac{H(\mathbf{x}) + H(\mathbf{y} | \mathbf{x}) - H(\mathbf{x})}{H(\mathbf{y})} = \frac{H(\mathbf{y} | \mathbf{x})}{H(\mathbf{y})} = \frac{H(\mathbf{y}) - (H(\mathbf{y}) - H(\mathbf{x} | \mathbf{y}))}{H(\mathbf{y})} \\ &= \frac{H(\mathbf{y}) - (H(\mathbf{y}) - H(\mathbf{x} | \mathbf{y}))}{H(\mathbf{y})} = 1 - \frac{I(\mathbf{x}; \mathbf{y})}{H(\mathbf{y})} \end{aligned}$$

and so

$$NCD_c(\mathbf{x}, \mathbf{y}) = 1 - \frac{I(\mathbf{x}; \mathbf{y})}{\max\{H(\mathbf{x}), H(\mathbf{y})\}}$$

■

The quantity $1 - \frac{I(\mathbf{x}; \mathbf{y})}{\max\{H(\mathbf{x}), H(\mathbf{y})\}}$ has been already proposed in [ferri2009experimental] as a candidate definition of the concept of Normalized Mutual Information. However, to the best of our knowledge, it has not been used in practice.

■ **Example 17.1**

■

R

In order to properly approximate the complexity of a continuous feature, we have to use a discretization algorithm that does not change the distribution of the observed values. For example, the kmeans algorithm, given that the optimization criteria is to minimize

In the nescience we use a uniform discretization, in which all the intervals have the same length. However, this approach is not optimal, especially when we are in a two dimensional space, in which we have to discretize the joint distribution of two attributes x and y .

For example, in the next figure, we can see a cloud of points, and how a uniform distribution has many intervals without points.

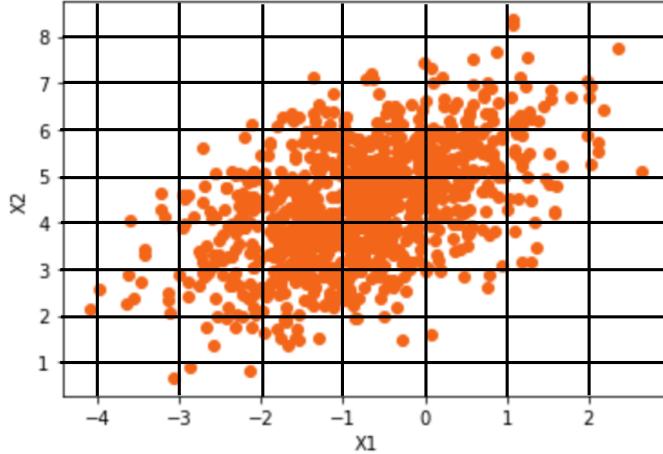


Figure 17.1: Uniform discretization.

A better approach would be to compute the convex hull around the actual cloud of points, and then divide the space into \sqrt{n} intervals of the same length, by for example, computing the centroid using a kmeans algorithm and then using voronoi polygons. However, up to the best of our knowledge, a discretization algorithm based on a constrained version of the kmeans in which the distribution of the points is not changed, has not been developed.

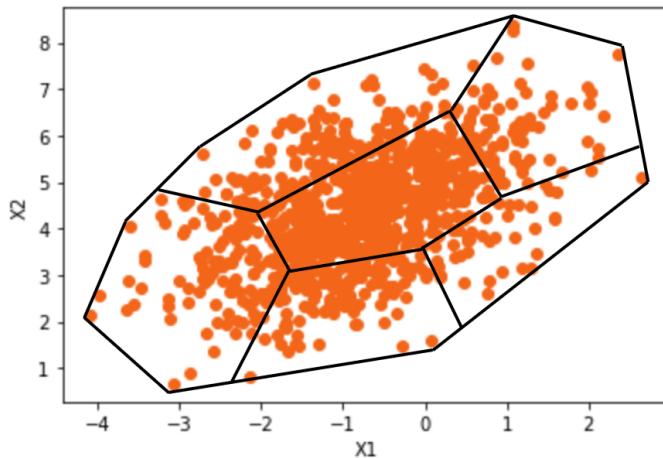


Figure 17.2: Convex hull and kmeans discretization.

17.3 Miscoding

In Section 10.1.1 we introduced the concept of miscoding as a quantitative measure of how well a string based encoding $r \in \mathcal{R}$ represents a research entity from \mathcal{E} . The miscoding of a representation

r was defined as:

$$\mu(r) = \min_{s \in \mathcal{R}_{\mathcal{E}}}^o \frac{\max\{K(s | r), K(r | s)\}}{\max\{K(s), K(r)\}}$$

and we saw that this quantity cannot be computed in practice for the general case. First of all because it requires a computation from an abstract oracle machine, second because it is based on the uncomputable Kolmogorov complexity, and third because it does not take into account the entity e in which we are interested.

In this section we are going to see how this concept can be adapted in practice to compute the error made by using a dataset \mathbf{X} as a representation of a response variable \mathbf{y} (see Section 7.2). Our goal is double, in one hand we are interested in measuring the quality of the dataset \mathbf{X} as a predictor of the variable \mathbf{y} , and in the other we want to identify those features \mathbf{x}_j of \mathbf{X} that have the higher predictive power for \mathbf{y} . This is the problem addressed by discriminative models (see Section 7.2.4) in which we want to estimate the conditional distribution $P(\mathbf{y} | \mathbf{X})$.

Given a training dataset \mathbf{X} , we can approximate the miscoding of a feature \mathbf{x}_j for the target variable \mathbf{y} by computing the normalized information distance between \mathbf{x}_j and \mathbf{y} (see Section 6.5):

$$E(\mathbf{x}_j, \mathbf{y}) = \frac{\max\{K(\mathbf{x}_j | \mathbf{y}), K(\mathbf{y} | \mathbf{x}_j)\}}{\max\{K(\mathbf{x}_j), K(\mathbf{y})\}}$$

The Kolmogorov complexity $K(\mathbf{v})$ of a vector \mathbf{v} will be approximated by the length of the compressed version of that vector $\hat{K}_C(\mathbf{v})$ using as compressor a minimal length code C (see Section 17.2).

Definition 17.3.1 Let \mathbf{y} be a response variable, \mathbf{X} a dataset composed by p features, and \mathbf{x}_j the j -th feature. We define the regular feature miscoding of \mathbf{x}_j as a representation of \mathbf{y} , denoted by $\hat{\mu}(\mathbf{x}_j, \mathbf{y})$, as:

$$\hat{\mu}(\mathbf{x}_j, \mathbf{y}) = \frac{\hat{K}_C(\mathbf{x}_j, \mathbf{y}) - \min\{\hat{K}_C(\mathbf{x}_j), \hat{K}_C(\mathbf{y})\}}{\max\{\hat{K}_C(\mathbf{x}_j), \hat{K}_C(\mathbf{y})\}}$$

Intuitively, the quantity $\hat{\mu}(\mathbf{x}_j, \mathbf{y})$ is a measure of the effort, as the length of a computer program and in relative terms, required to fully encode \mathbf{y} assuming a knowledge of \mathbf{x}_j , and the other way around. The lower this value, the better would be the quality of \mathbf{x}_j as a predictor for \mathbf{y} .

■ **Example 17.2** Let's \mathbf{y} be a target variable composed by 1.000 random samples that follows a normal distribution $N(3, 1)$ with mean $\mu = 3$ and standard deviation $\sigma = 1$, \mathbf{x}_1 be a predictor feature that is equal to \mathbf{y} with some random noise, that is $\mathbf{x}_1 = \mathbf{y} + N(3, 1)/10$, and \mathbf{x}_2 be a second predictor based on random samples from a exponential distribution with a rate of $\lambda = 1$.

```
from scipy.stats import norm, expon

y = norm.rvs(loc=3, scale=1, size=10000)
x1 = y + norm.rvs(loc=3, scale=1, size=10000) / 10
x2 = expon.rvs(size=10000)
```

We can use the Nescience library to compute the miscoding of the features \mathbf{x}_1 and \mathbf{x}_2 when they encode the target variable \mathbf{y} .

```
from fastautoml.miscoding import Miscoding
import numpy as np

X = np.column_stack((x1, x2))
```

```
miscoding = Miscoding()
miscoding.fit(X, y)
miscoding.miscoding_features(mode="regular")
```

The output of the library would be something similar to the following²:

```
array([0.27445364, 0.9934222])
```

As it was expected the miscoding of $\hat{\mu}(\mathbf{x}_1, \mathbf{y})$ is much smaller than the miscoding of $\hat{\mu}(\mathbf{x}_2, \mathbf{y})$. In this case, we should prefer \mathbf{x}_1 over \mathbf{x}_2 as a predictor of \mathbf{y} .

Sometimes we will use the normalized version of the complements of the individual miscodings, that is $\frac{1-\hat{\mu}(\mathbf{x}_i, \mathbf{y})}{\sum_{j=1}^p 1-\hat{\mu}(\mathbf{x}_j, \mathbf{y})}$, instead of the regular ones $\hat{\mu}(\mathbf{x}_i, \mathbf{y})$, because they are easier to compare with other feature selection techniques, and because they have a visually appealing interpretation. We call this version of miscoding the *adjusted* feature miscoding.

■ **Example 17.3** In this example we are going to generate a synthetic dataset where the target variable \mathbf{y} is a collection of normally-distributed clusters of points, and the training set \mathbb{X} is composed by both, relevant and irrelevant predictors. In particular we will generate 1.000 samples composed by 20 features that describe 10 clusters; only 4 of the features are relevant for prediction, and the other remaining 6 are just random values.

In Figure 17.3 we can see a two-dimensional projection of this dataset, along the hyperplane composed by features 8 and 10.

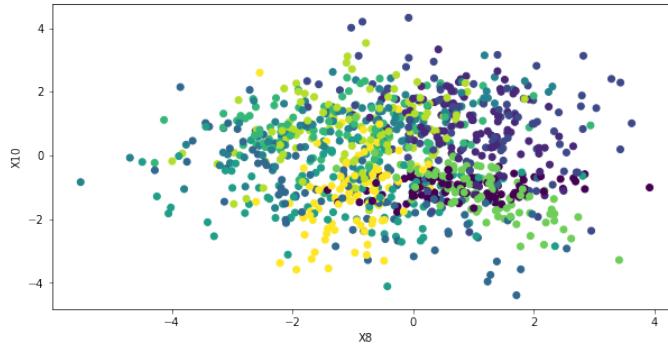


Figure 17.3: Gaussian Blob Cluster.

```
from fastautoml.miscoding import Miscoding
from sklearn.datasets.samples_generator import make_classification

X, y = make_classification(n_samples=1000, n_features=20, n_informative=4,
                           n_redundant=0, n_classes=10, n_clusters_per_class=1, flip_y=0)

miscoding = Miscoding()
miscoding.fit(X, y)
msd = miscoding.miscoding_features(mode='adjusted')
```

We will use the adjusted version of the miscoding for an easier comparison with other feature selection techniques. If we plot the results (see Figure 17.4) we will see that the library has successfully identified the four relevant predictors (\mathbf{x}_3 , \mathbf{x}_8 , \mathbf{x}_{10} and \mathbf{x}_{16}). Since we are using the adjusted version of miscodings, the higher the value the better, and mind that actual values have to be interpreted in relative terms.

²Since we are generating a list of 1.000 random samples, the reader could get a slightly different result when running this example.

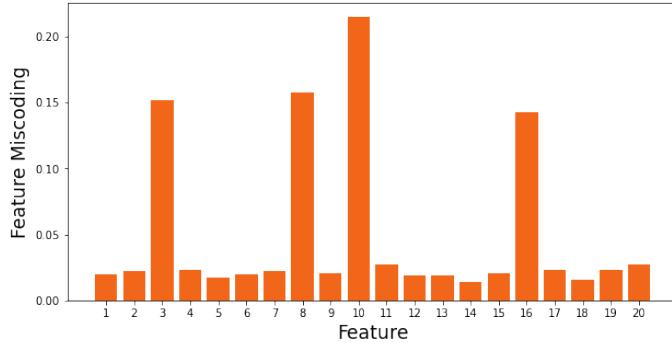


Figure 17.4: Miscoding of a Synthetic Dataset.

We can compare miscoding with correlation, a common technique used in machine learning to identify the most relevant features of a dataset. In Figure 17.5 is shown correlation between the individual features that compose \mathbf{X} and the target variable \mathbf{y} . As we can observe, correlation fails to properly identify one of the relevant features (\mathbf{x}_3).

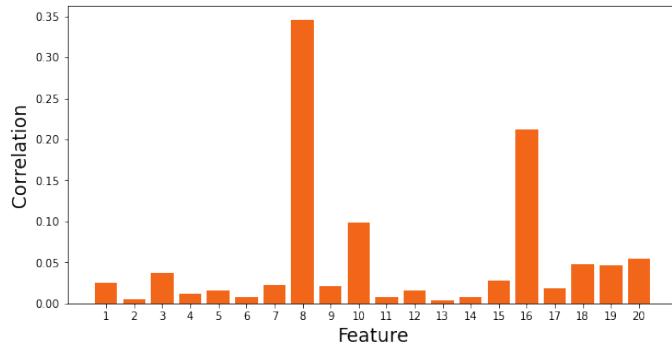


Figure 17.5: Correlation of a Synthetic Dataset.

Feature miscoding allow us to identify the most relevant features of a training dataset \mathbf{X} , but it cannot be used to compute the miscoding of the dataset itself. If we start with a miscoding of 1 (full unknown), and subtract the miscodings of the individual features, we will end up with a negative miscoding, something that it is not allowed by our theory. If we use the adjusted version, the dataset miscoding will be 0 for all datasets, which is against our intuition that not all possible datasets \mathbf{X} represent equally well a target variable \mathbf{y} . According to the theory of nescience, we expect that non-relevant features add, instead of subtract, to the global miscoding of the dataset.

In order to address this problem, we have to introduce the concept of partial miscoding of a feature, as the difference between the adjusted and normalized miscodings.

Definition 17.3.2 Let \mathbf{y} be a target variable, \mathbf{X} a dataset composed by p features, and \mathbf{x}_j the j -th feature. We define the partial miscoding of \mathbf{x}_j as a representation of \mathbf{y} , denoted by $\hat{\mu}(\mathbf{x}_j, \mathbf{y})$, as:

$$\hat{\mu}(\mathbf{x}_i, \mathbf{y}) = \frac{1 - \hat{\mu}(\mathbf{x}_i, \mathbf{y})}{\sum_{j=1}^p 1 - \hat{\mu}(\mathbf{x}_j, \mathbf{y})} - \frac{\hat{\mu}(\mathbf{x}_i, \mathbf{y})}{\sum_{j=1}^p \hat{\mu}(\mathbf{x}_j, \mathbf{y})}$$

A positive partial miscoding means that the feature contributes to describe the target variable, meanwhile a negative value means that the feature is not relevant.

■ **Example 17.4** We will use again the synthetic dataset of Example 17.3, but we will increase the number of relevant features from 4 to 14. Then, we will compute the list of partial miscodings.

```
from fastautoml.miscoding import Miscoding
from sklearn.datasets.samples_generator import make_classification

X, y = make_classification(n_samples=1000, n_features=20, n_informative=14,
                           n_redundant=0, n_classes=10, n_clusters_per_class=1, flip_y=0)

mCoding = Miscoding()
mCoding.fit(X, y)
msd = mCoding.miscoding_features(mode="partial")
```

As we can see in Figure 17.6, not only the library has been able to correctly identify the relevant features, but also, non relevant features have now a negative contribution to the global miscoding.

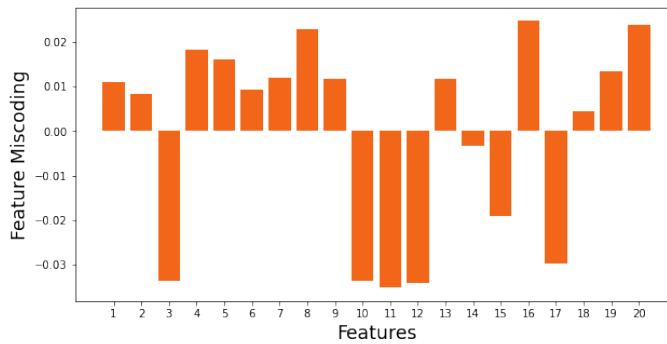


Figure 17.6: Partial Feature Miscoding.

Given the definition of partial feature miscoding we can provide a definition of the concept of miscoding of a target variable given a subset of predictors that it is closer to the original concept of miscoding defined by the theory of nescience.

■ **Definition 17.3.3** Let \mathbf{y} be a target variable, $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$ a dataset composed by p features, and $\mathbf{Z} = \{\mathbf{z}_1, \dots, \mathbf{z}_k\}$ a subset of features, that is, $\{\mathbf{z}_1, \dots, \mathbf{z}_k\} \subseteq \{\mathbf{x}_1, \dots, \mathbf{x}_p\}$. We define the miscoding of \mathbf{Z} as a representation of \mathbf{y} , denoted by $\hat{\mu}(\mathbf{Z}, \mathbf{y})$, as:

$$\hat{\mu}(\mathbf{Z}, \mathbf{y}) = \sum_{i=1}^k \tilde{\mu}(\mathbf{z}_i, \mathbf{y})$$

■ **Example 17.5** Based on the dataset and the partial features miscoding computed in Example 17.4, in Figure 17.7 we can see the evolution of the miscoding of the training subset \mathbf{Z} as we add more features to the study.

In the following example we are going to compare the performance of a machine learning classifier when using a full dataset and a reduced version of the same dataset using only those features identified as relevant, i.e., with positive partial miscoding.

■ **Example 17.6** Let's train a neural network with the standard MNIST dataset in order to classify hand written digits. The evaluation criteria will be the score of the classifier, that is, the percentage of digits correctly classified, applied over a test dataset different from the dataset used for training. The neural network will be trained and evaluated using all the features that compose the dataset, and with a reduced version of the dataset composed by only those features with a positive partial

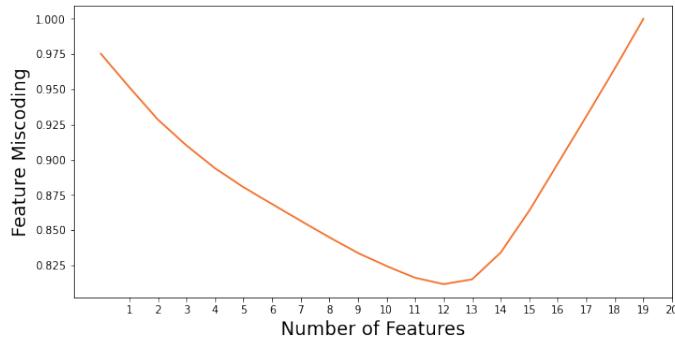


Figure 17.7: Accumulated Partial Feature Miscoding.

miscoding.

```

import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import load_digits
from fastautoml.fastautoml import Miscoding

data = load_digits()
X_raw = data.data
y_raw = data.target

mCoding = Miscoding()
mCoding.fit(X_raw, y_raw)
mscd = mCoding.miscoding_features(misCoding='partial')
X_red = X_raw[:, np.where(mscd > 0)[0]]
y_red = y_raw

X_raw_train, X_raw_test, y_raw_train, y_raw_test = train_test_split(X_raw,
                                                                    y_raw, test_size=.3)
X_red_train, X_red_test, y_red_train, y_red_test = train_test_split(X_red,
                                                                    y_red, test_size=.3)

clf = MLPClassifier(alpha=1, max_iter=1000)

clf.fit(X_raw_train, y_raw_train)
score_raw = clf.score(X_raw_test, y_raw_test)

clf.fit(X_red_train, y_red_train)
score_red = clf.score(X_red_test, y_red_test)

reduction = 1 - X_red_train.shape[1] / X_raw_train.shape[1]

print("Score raw:", score_raw, " Score MisCoding:", score_red,
      " Reduction:", reduction)

Score raw: 0.9833333333333333  Score MisCoding: 0.9814814814814815  Data Reduction: 0.46875

```

If we run the above source code, we will see that the score of the neural network classifier is about the same for the two datasets, 98% of the digits are correctly classified using the test data. However, the reduced dataset used for training based on the optimal miscoding is 43% smaller than the original dataset. This size reduction could have a big impact in the training time of the neural network. Smaller datasets are also relevant when working with ensembles of models, like random forests, where hundreds or thousands of models have to be trained.

Intuitively, as Example 17.5 shows, we should prefer the subset \mathbf{Z} of \mathbf{X} composed by all those features whose partial miscoding are greater than zero. However, as we will see in the following

sections of this chapter, this might not be the case. Feature selection is only one of the criteria used in the process of finding an optimal model for an entity represented by a dataset. It might happen that other elements, like inaccuracy or surfeit, suggest to use a different subset of predictors. The global optimization criteria we should use is the concept of nescience. A sensible approach to use partial miscoding would be to incrementally add to our model those features with higher miscoding, until all features with a positive value have been added, or an optimality criterion has been reached.

In case of having a generative model (see Section 7.2.4), that is, a machine learning algorithm designed to find the joint probability $P(\mathbf{X}, \mathbf{y})$, we could use miscoding to compute how the different features relate to each other, that is, the quantity $\hat{\mu}(\mathbf{x}_i, \mathbf{x}_j)$ for each pair of values $i, j \leq p$. The result would be a miscoding matrix (see Example 17.7).

■ Example 17.7 The Boston dataset included in `scikit-learn` library contains a collection of variables that (potentially) could explain the price of houses in the area of Boston. In this example, instead of computing which are the factors that contribute the most to the price of houses, we are going to study the inter-dependence between these factors, using a miscoding matrix.

```
from fastautoml.fastautoml import Miscoding
from sklearn.datasets import load_boston

data = load_boston()

miscoding = Miscoding(X_type="numeric", y_type="numeric")
miscoding.fit(data.data, data.target)
mscd_matrix = miscoding.features_matrix(mode='regular')
```

In Figure 17.8 we can see a graphical representation using a heatmap of the miscoding matrix computed over the features. The darker values represent a lower miscoding (mind we are using the regular version of the concept of miscoding). In particular, the values of the main diagonal are equal to zero.

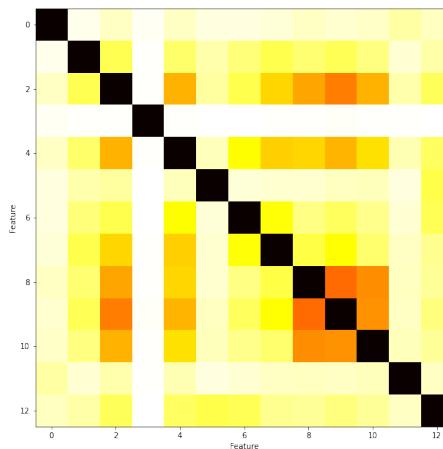


Figure 17.8: Regular Miscoding Matrix.

The minimum value of 0.52 is obtained with the pair (8, 9) that correspond to the features "index of accessibility to radial highways" and "full-value property-tax rate per \$10,000". These features are good candidates to evaluate in a predictive model, since they contain non-redundant information. The maximum value of 0.99 is achieved with the pair (3, 12) with the features "Charles River dummy variable" and "% lower status of the population". These features contains almost the same information, and so, including both in a model does not add nothing new, but increases the complexity of the model and the risk of over-fitting.

17.4 Inaccuracy

In Section 11.1 we defined the inaccuracy of a description $d \in \mathcal{D}$ for a representation $r \in \mathcal{R}$ as the normalized information distance between the representation r and the string $\Gamma(d)$ printed out by a universal Turing machine when given the description as input:

$$\iota(d, r) = \frac{\max\{K(r | \Gamma(d)), K(\Gamma(d) | r)\}}{\max\{K(r), K(\Gamma(d))\}}$$

Inaccuracy, being based in Kolmogorov complexity, is not computable for the general case, and so, it has to be approximated in practice. In this section we are going to see how this concept can be estimated in case of a model trained using a dataset. The approach will be similar to the one used in case of miscoding (see Section ?? for more information).

TODO: This definition correspond to the discriminative case. Introduce the generative case as well.

Definition 17.4.1 Let \mathbb{X} be a dataset, \mathbf{y} a response variable, m a model, and $\hat{\mathbf{y}} = m(\mathbb{X})$ the predicted values by m given \mathbb{X} . We define the *inaccuracy* of the model m for the target values \mathbf{y} , denoted by $\hat{\iota}(\hat{\mathbf{y}}, \mathbf{y})$, as:

$$\hat{\iota}(\hat{\mathbf{y}}, \mathbf{y}) = \frac{\hat{K}_C(\hat{\mathbf{y}}, \mathbf{y}) - \min\{\hat{K}_C(\hat{\mathbf{y}}), \hat{K}_C(\mathbf{y})\}}{\max\{\hat{K}_C(\hat{\mathbf{y}}), \hat{K}_C(\mathbf{y})\}}$$

Intuitively, the quantity $\hat{\iota}(\hat{\mathbf{y}}, \mathbf{y})$ is a measure of how far are the predicted values from real values. The lower this quantity, the better is the quality of m as a predictor for \mathbf{y} . With our new inaccuracy metric we are measuring not only how difficult is to reconstruct the original target vector \mathbf{y} given the predicted values $\hat{\mathbf{y}}$, but also how much additional information $\hat{\mathbf{y}}$ contains that is not related to \mathbf{y} , being the latter a novelty with respect to other metrics used in machine learning to measure the accuracy of a model.

■ **Example 17.8** Inaccuracy, according to the minimum nescience principle, is given by the normalized compression distance between the actual targets \mathbf{y} and the predicted targets $\hat{\mathbf{y}}$ by the model. In the following example we are going to compare the behavior of our new inaccuracy metric with a classical score metric. The experiment will be based on the MNIST dataset (hand written digits recognition) provided by scikit-learn.

```
from fastautoml.fastautoml import Inaccuracy
from sklearn.datasets import load_digits

X, y = load_digits(return_X_y=True)

inacc = Inaccuracy()
inacc.fit(X, y)
```

For this example we will train a decision tree classifier up to a pre-determined tree depth of i , where i goes from 1 to 20.

```
from sklearn.tree import DecisionTreeClassifier

scores      = list()
inaccuracies = list()

for i in range(20):

    tree = DecisionTreeClassifier(max_depth=i, random_state=42)
    tree.fit(X, y)

    scores.append(1 - tree.score(X, y))
    inaccuracies.append(inacc.inaccuracy_model(tree))
```

We are interested to compare the behavior of score (actually we are comparing against one minus score) and inaccuracy metrics. As we can see in Figure 17.9, both metrics present a similar behavior, having inaccuracy a larger value, due to a stronger emphasis in incorrectly predicted values.

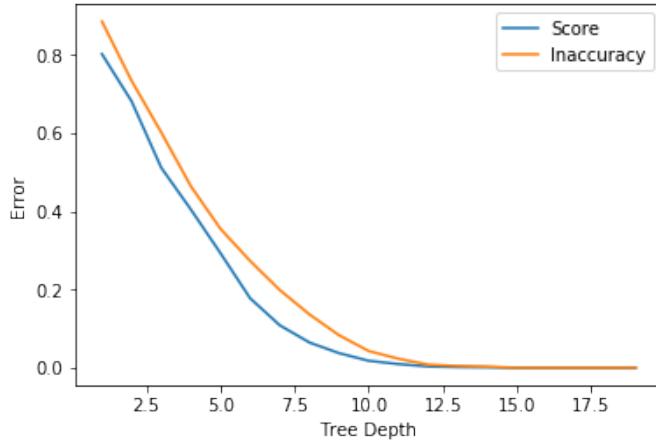


Figure 17.9: Inaccuracy vs. Score of Decision Trees

In Example 17.8 we have seen that the deeper the tree, the smaller is the training error. Of course, the higher the value of i , the higher the risk of overfitting the data. However, in case of inaccuracy we are not interested in avoiding overfitting, since overfitting is controlled by the metric of surfeit (see Section ??).

We can see inaccuracy as the effort, measured as the length of a computer program, required to fix the predictions made by a model. In this sense, according to the minimum nescience principle, it is not the same a model that makes one hundred times the same error than a model that makes one hundred different errors, since it should be easier to fix the former than the later (see Example 17.9).

■ **Example 17.9** In this example we are going to use again a decision tree classifier, but this time it will be trained with the hyperparameter minimum number of samples per leaf node set to 5 (a common approach used in practice to avoid decision trees to overfit).

```
tree = DecisionTreeClassifier(min_samples_leaf=5)
tree.fit(X, y)
```

The inaccuracy of this new trained model is 0.17, and its score 0.08. Next we will artificially introduce one hundred errors in the dataset, simulating the case that the tree is not able to model correctly these data points. In this particular case all the errors are exactly the same.

```
X2 = X.copy()
y2 = y.copy()
for i in range(100):
    X2 = np.append(X2, [X[0]], axis=0)
    y2 = np.append(y2, (y[0]+1) % 10)
```

The inaccuracy of the decision tree, given this new dataset, has increased³ from 0.17 to 0.21.

³Note that we had to `fit()` again the class `Inaccuracy` in order to use the new dataset. Normally this is not the way we use this class; instead what we should do is to fit once a dataset, and then compute the inaccuracy of different models. We are doing here in this way to demonstrate an interesting property of the concept of inaccuracy.

```
inacc.fit(X2, y2)
inacc.inaccuracy_predictions(pred)
```

Score has also increased, in this case from 0.08 to 0.13.

```
1 - tree.score(X2, y2)
```

Finally, we are going to repeat exactly the same experiment, but this time instead of adding one hundred times the same error, adding one hundred different errors.

```
X3 = X.copy()
y3 = y.copy()
for i in arange(100):
    index = np.random.randint(X.shape[0])
    X3    = np.append(X3, [X[index]], axis=0)
    y3    = np.append(y3, (y[index]+1) % 10)
```

In this last case the inaccuracy of the model has increased up to 0.25, meanwhile score remained the same. ■

In line with Example 17.9, an extreme case would be a model for a target binary variable (True and False) that always fails with its predictions, that is, if the value of the target is True, the model will predict False, and if it is False, it will predict True. The classical evaluation metrics would say that this model is the worst possible model, but our inaccuracy would claim that the model is perfect. We might be wondering what it is the value of a model that always fails to predict the correct target. But if we are the managers of an edge fund investing in the stock market, we will very happy to pay a huge amount of money for a model that predicts that the shares of IBM will go down whenever they go up, and the other way around.

In case of having a highly unbalanced dataset, that is, when some categories have a lot of more training data than others, the classical score metric can provide a misleading result, since a good score does not necessarily mean a good model, it might happen that the model is simply properly classifying the samples of the category with the higher number of training samples, and misclassifying the others. In practice, we solve this problem by using metrics specifically designed to deal with unbalanced datasets. In case of the new metric of inaccuracy, as Example 17.10 shows, a model that can not properly classify one of the categories is considered a bad model, even if this category has only a few points in the training dataset.

■ **Example 17.10** For this example, we will create a synthetic dataset using the `make_classification` utility of scikit-learn, with two classes in which one of them has 95% of the samples, and the other 5%.

```
from sklearn.datasets import make_classification

depth = list()
score = list()
inacc = list()

inaccuracy = Inaccuracy()

for i in np.arange(1, 100):

    X, y = make_classification(n_samples=1000, n_features=2,
                               n_informative=2, n_redundant=0,
                               class_sep=2, flip_y=0, weights=[0.95,0.05])

    inaccuracy.fit(X, y)

    tree = DecisionTreeClassifier(min_samples_leaf=i)
    tree.fit(X, y)
```

```

depth.append(i)
score.append(1 - tree.score(X, y))
inacc.append(inaccuracy.inaccuracy_model(tree))

```

The experiment consists in training a decision tree classifier with a minimum number of samples per leaf of i , where i goes from 1 to 100. In Figure 17.10 we can see the behavior of inaccuracy and score. In case of large values of i , the score metric tell us that no more than a 5% of the samples is misclassified, however, the inaccuracy says that even if the total number of misclassified points is low, the inaccuracy of the model is very bad.

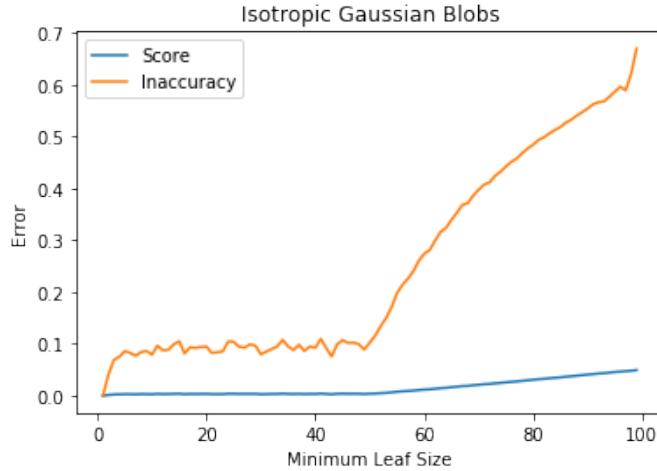


Figure 17.10: Inaccuracy of Decision Tree.

■

17.5 Surfeit

In Section 12.1 we defined the surfeit of the model $m \in \mathcal{M}$ for the representation $r \in \mathcal{R}$ as:

$$\sigma(m, r) = 1 - \frac{K(r)}{l(m)}$$

Since the length $K(r)$ of shortest possible model for the representation r is in general unknown, we have to approximate this concept in practice. In case of having a training dataset \mathbb{X} and a target variable \mathbf{y} , we can approximate the surfeit of a model m for the representation \mathbf{y} by means of computing:

$$\hat{\sigma}(m, \mathbf{y}) = 1 - \frac{\hat{K}_C(\mathbf{y})}{l(m)}$$

Where $\hat{K}_C(\mathbf{y})$ is the length of the compressed version of the vector \mathbf{y} using as compressor a minimal length code C , computed given the relative frequencies of the values observed in \mathbf{y} (see Section ??).

Definition 17.5.1 Let \mathbf{y} be a response variable, \mathbb{X} a dataset composed by p features and n samples. We define the surfeit of the model $m \in \mathcal{M}$ as a representation of \mathbf{y} , denoted by $\hat{\sigma}(m, \mathbf{y})$,

as:

$$\hat{\sigma}(m, \mathbf{y}) = 1 - \frac{\hat{K}_C(\mathbf{y})}{l(m)}$$

The definition of surfeit requires a method of encoding the models as a string of symbols, so we can compute their length. Ideally, we should use as encodings Turing machines, and agree upon an universal Turing machine to interpret those models. However, that would make very difficult to add new models to the nescience library. Instead, we have used for the encoding of models a simplified version of the Python language, where not all the constructions are allowed, and we do not allow the use of libraries.

Surfeit is a metric that can help us to avoid overfitted models. The higher is the surfeit of a model, the higher is the probability that the model is an overfit of the training dataset, as Example 17.11 shows.

■ **Example 17.11** In this example we are going to generate a dataset composed by 900 samples of a sinusoidal curve, and we will fit the data using a n degree polynomial, where n goes from 1 to 15.

```
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

from Nescience.Nescience import Surfeit
from Nescience.Nescience import Inaccuracy

n_samples = 900
degrees = np.arange(1, 15)

X = np.sort(np.random.rand(n_samples) * 3)
y = np.cos(1.5 * np.pi * X)

linacc = list()
lsurfeit = list()

for i in degrees:

    poly = PolynomialFeatures(degree=i, include_bias=False)
    newX = poly.fit_transform(X[:, np.newaxis])

    linear_regression = LinearRegression()
    linear_regression.fit(newX, y)

    inacc.fit(newX, y)
    inaccuracy = inacc.inaccuracy_model(linear_regression)

    sft.fit(newX, y)
    surfeit = sft.surfeit_model(linear_regression)

    linacc.append(inaccuracy)
    lsurfeit.append(surfeit)
```

In figure 17.11 we can see the results of this experiment. As it was expected, the higher the degree of the polynomial, the smaller is the error of the model. However, at the same time we see that the higher the polynomial, the higher the surfeit of the model. The ideal model is that one that has a low inaccuracy and a low surfeit.

Another advantage of the concept of surfeit is that it allows us to compare and decide between models that belong to different families. For example, in case of models having the same accuracy, shall we prefer a decision tree over a neural network, or a naive Bayes classifier over a support vector machine? Next example shows how we can decide about those questions.

■ **Example 17.12** In this example we are going to compare a decision tree with a neural network.

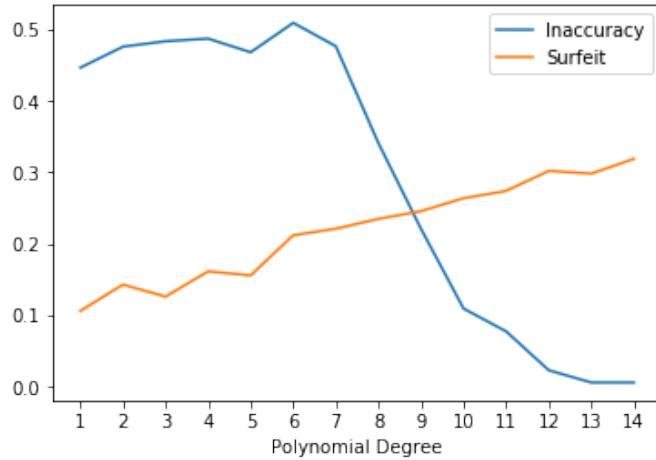


Figure 17.11: Surfeit vs Inaccuracy

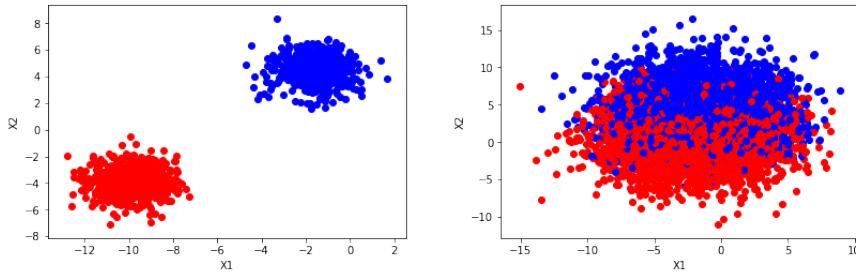


Table 17.1: Isotropic Gaussian blobs.

We will use a synthetic dataset composed by two isotropic Gaussian blobs, and we will train our models to split them apart. In the first part of the example we will use a standard deviation of 1 and only two dimensions, so the two clusters are easy to classify (see figure on Table 17.1, left side).

```

from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from Nescience.Nescience import Surfeit
from Nescience.Nescience import Inaccuracy
from sklearn.datasets.samples_generator import make_blobs

X, y = make_blobs(n_samples=1000, centers=2, n_features=2, cluster_std=1)

tree = DecisionTreeClassifier()
tree.fit(X, y)
tree.score(X, y)

nn = MLPClassifier()
nn.fit(X, y)
nn.score(X, y)

sft = Surfeit()
sft.fit(X, y)

sft.surfeit_model(tree)
sft.surfeit_model(nn)

```

If we ran the above code we will see that both models have exactly the same accuracy of 1,

that is, they are perfect classifiers. However the surfeit of the decision tree is 0.25, meanwhile the surfeit of the neural network is 0.73. In this particular case we should prefer the decision tree over the neural network.

If we perform the same experiment using a standard deviation of 3, so two clusters that are more difficult to split (see Table 17.1, right side), the situation will change.

```
X, y = make_blobs(n_samples=10000, centers=2, n_features=8, cluster_std=3)
```

In this second case, again, both models have the same accuracy (we have increased the number of samples, and the number of dimensions, so the models can still perform a perfect classification), but the surfeit of the decision tree has increased to 0.82, and the surfeit of the neural network is almost the same, 0.76. For this second dataset we should prefer the neural network over the decision tree.

In example 17.12 we have assumed that both models, decision tree and neural networks, have the same accuracy. When this is not the case, when the models do not have the same accuracy, we have to apply to the concept of nescience in order to decide between them.

 **TODO:** Mention the problem of the stability of the signal.

17.6 Nescience

In Chapter 14 we defined the concept of nescience as the solution to a non-linear multi-objective optimization problem, where we had to minimize the miscoding, inaccuracy and surfeit of representations and models. The solution to this problem is, in general, not unique, in the sense that we can find multiple pairs of representations and models that have the property that we can not improve one of these quantities without degrading the others (Pareto optimality). However, in practice, we expect that a machine learning library should provide a single solution when training a model over a dataset. In order to provide this unique solution, we have to resort to a utility function that selects one from the available solutions. The nescience library provide different alternatives of utility functions, being the default one the arithmetic mean the tree metrics.

 The nescience library implements the following utility functions to approximate the concept of nescience, that is, to compute $\hat{V}(\mathbb{Z}, m, \mathbf{y})$:

- Euclid distance: $(\hat{\mu}(\mathbb{Z}, \mathbf{y})^2 + \hat{i}(\hat{\mathbf{y}}, \mathbf{y})^2 + \hat{\sigma}(m, \mathbf{y})^2)^{1/2}$
- Arithmetic mean: $\frac{\hat{\mu}(\mathbb{Z}, \mathbf{y}) + \hat{i}(\hat{\mathbf{y}}, \mathbf{y}) + \hat{\sigma}(m, \mathbf{y})}{3}$
- Geometric mean: $(\hat{\mu}(\mathbb{Z}, \mathbf{y}) \times \hat{i}(\hat{\mathbf{y}}, \mathbf{y}) \times \hat{\sigma}(m, \mathbf{y}))^{1/3}$
- Product: $\hat{\mu}(\mathbb{Z}, \mathbf{y}) \times \hat{i}(\hat{\mathbf{y}}, \mathbf{y}) \times \hat{\sigma}(m, \mathbf{y})$
- Addition: $\hat{\mu}(\mathbb{Z}, \mathbf{y}) + \hat{i}(\hat{\mathbf{y}}, \mathbf{y}) + \hat{\sigma}(m, \mathbf{y})$
- Weighted mean: $w_\mu \hat{\mu}(\mathbb{Z}, \mathbf{y}) + w_i \hat{i}(\hat{\mathbf{y}}, \mathbf{y}) + w_\sigma \hat{\sigma}(m, \mathbf{y})$
- Harmonic mean: $\frac{3}{\hat{\mu}(\mathbb{Z}, \mathbf{y})^{-1} + \hat{i}(\hat{\mathbf{y}}, \mathbf{y})^{-1} + \hat{\sigma}(m, \mathbf{y})^{-1}}$

Euclid distance and addition have the drawback that they produce nescience values greater than one, something that it is against our theory. Geometric mean, product and harmonic mean have the problem that the nescience is zero, or not defined, if one of the three metrics (miscoding, inaccuracy or surfeit) is zero. And the weighted mean introduce three new hyperparameters that have to be optimized. It is still an open question which one is the best utility function to compute the nescience of a dataset and a model.

Example 17.13 shows how we can use the nescience library to compute the nescience of a dataset and a model.

■ **Example 17.13** This example shows how to compute in practice the nescience of a dataset and a model. In particular, we are going to compute the nescience of a decision tree classifier applied over the dataset digits (MNIST hand written digits classification problem) included in the `sklearn` library.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_digits
from Nescience.Nescience import Nescience

data = load_digits()

tree = DecisionTreeClassifier()
tree.fit(data.data, data.target)
tree.score(data.data, data.target)
[ ] 1

nescience = Nescience()
nescience.fit(data.data, data.target)

nescience.nescience(tree)
[ ] 0.5895603819965907
```

The score of the decision tree model is 1, meaning that all the samples have been properly classified. Of course, what happened is that the decision tree is overfitting the dataset. In order to avoid this kind of problems we usually split the data in separate training and testing subsets, or we perform a more advanced cross-validation. However, if we compute the nescience, we will get a value of 0.59, rising the flag that something is wrong with the model or the training dataset.

In Example 17.13 we have shown that one of the advantages of the concept of nescience is that we can evaluate the quality of a model without applying computationally expensive procedures like cross-validation, and without requiring to save part of the data as a test subset. Another advantage of the metric nescience is that it allows us to decide between competing models from different families of models, as it is shown in Example 17.14.

■ **Example 17.14** In this example we are going to compare two models from two different families of models: decision trees and neural networks. Both models will be trained with the breast cancer dataset provided by the `sklearn` library.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import load_breast_cancer
from Nescience.Nescience import Nescience

data = load_breast_cancer()
X = data.data
y = data.target

tree = DecisionTreeClassifier(max_depth=3)
tree.fit(X, y)
tree.score(X, y)
[ ] 0.9789103690685413

nescience = Nescience()
nescience.fit(X, y)
nescience.nescience(tree)
[ ] 0.5945936419010083

nn = MLPClassifier()
nn.fit(X, y)
nn.score(X, y)
[ ] 0.9261862917398945
```

```
nescience.nescience(nn)
[ ] 0.7860523786210711
```

Both models have a similar score. In this case, not only the decision tree provide a better score, but also, the nescience is much lower than in case of the multi-layer perceptron, and so, we should prefer the former over the later.

Nescience is a metric that can be used to optimize the hyperparameters that define a (parametric) family of models. The advantage of nescience is that we can use a greedy approach to select the best value for an hyperparameter, saving a lot of computational time and resources during the search. That is, if we have a model controlled by an hyperparameter such that the higher the value the better the score, we should select that value in which the nescience stops decreasing and starts to increase, since this is the point in which we are not longer learning anything new from that dataset (see Example 17.15).

■ **Example 17.15** For this example we will use again a decision tree classifier with the breast cancer dataset. We will train 10 different trees, setting the hyperparameter `max_depth` with values from 1 to 10. The `max_depth` hyperparameter controls how deep we allow the tree to grow in order to classify the samples of the dataset. The deeper the tree the higher the score of the model, but also, the higher the risk of overfitting the training data. For each tree we will compute the nescience of the model, and we will compare it with a cross validation score. The results are shown in Figure 17.12. As we can see in the figure, both, nescience and cross validation score, decrease as we increase the depth of the tree, until we reach a point in which it starts to increase. This inflection point is where the model begins to overfit the data. The nescience library suggests to use a tree with a maximum depth of 7, meanwhile with the cross validation we got an optimal level of 6.

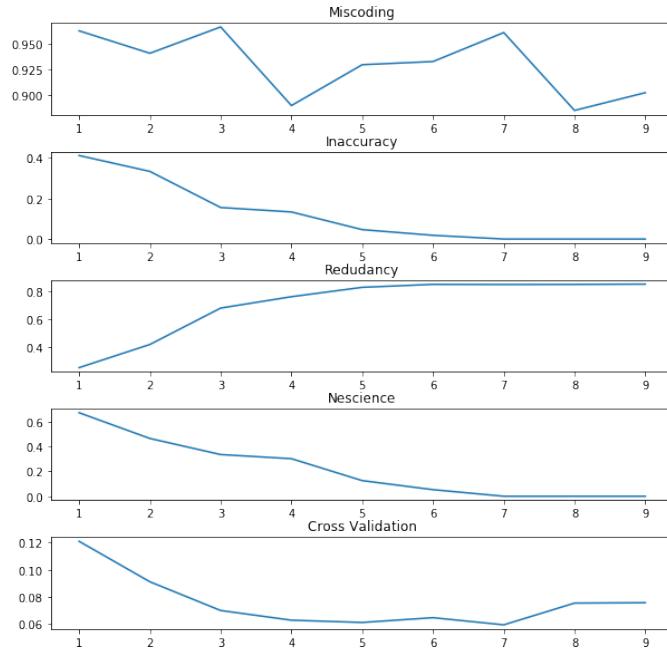


Figure 17.12: Evolution of Nescience with Tree Depth.

It is interesting to note the behavior of the three metrics that define the concept of nescience in Figure 17.12. As it is expected the the deeper the tree the smaller is the inaccuracy of the model and the higher the surfeit. However, in case of miscoding, we have a sort of random evolution. This

behavior is due to the fact that each candidate tree uses a different subset of features at the decision nodes. I would be very nice to have an decision tree building algorithm that takes into account miscoding in order to decide the best features for new branches. Such an algorithm is described in Section 17.11.

Finally, we are going to see how to use nescience in case of hyperparameter searches where we cannot apply a greedy approach, for example when the search is performed over a collection of (usually conflicting) hyperparameters. Hyperparameter search is a computationally expensive approach, since the number of possible combinations to test could be very large. Moreover, if for each candidate set we have to cross-validate the result, the search becomes prohibitive. As we have seen, nescience do not requires the use of crossvalidation to detect a situation of overfitting, and so, it can significantly speed up the process of searching for optimal hyperparameters. In Example 17.16 it is show how we can do that with the nescience library.

■ **Example 17.16** In this example we are going to see how we can use the nescience library to find the optimal hyperparameters for a model using a grid search. In particular, we are going to select the best hyperparameters for a multilayer perceptron classifier, including the number of hidden layers, and the size of those layers (what it is called Neural Architecture Search). The procedure will be demonstrated using the digits dataset.

```
from Nescience.Nescience import Nescience
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
```

First of all we have to provide a custom loss function based on the concept of nescience to be integrated with the search procedure. The next code shows how to implement such a function.

```
def my_custom_loss_func(estimator, X, y):
    nsc = Nescience()
    nsc.fit(X, y)
    nescience = nsc.nescience(estimator)

    # scikit-learn expect that higher numbers are better
    score = -nescience

    return score
```

Second, we have to define the grid of hyperparameters over which we are going to do the search. The larger the grid, the better the result, but also, the more computer time is required to evaluate all possible combinations.

```
parameters = {'solver': ['lbfgs'],
              'max_iter': [1000, 1500, 2000],
              'alpha': 10.0 ** -np.arange(1, 10, 3),
              'hidden_layer_sizes': [(60,), (100,), (60, 60,), (100, 100,), (60, 60, 60,), (100, 100, 100,)]}
```

Next code show how to do a classical grid search using the score of the models. The search will be evaluated using a train/test split of the dataset.

```
clf_std = GridSearchCV(estimator=MLPClassifier(), param_grid=parameters,
                      cv=3, iid=True, n_jobs=-1)
clf_std.fit(X_train, y_train)
clf_std.best_params_
[] {'alpha': 0.1,
[] 'hidden_layer_sizes': (100,),
[] 'max_iter': 1000,
```

```
[ ] 'solver': 'lbfgs'}

y_true, y_pred = y_test, clf_std.predict(X_test)
print(classification_report(y_true, y_pred))

[] precision    recall   f1-score   support
[] avg / total       0.98       0.97       0.97      540
```

Next code show how to perform exactly the same search, but using the concept of nescience instead of the metric score.

```
clf_nsc = GridSearchCV(estimator=MLPClassifier(), param_grid=parameters,
                       cv=3, scoring=my_custom_loss_func, iid=True)
clf_nsc.fit(X_train, y_train)
clf_nsc.best_params_

{'alpha': 0.1,
 'hidden_layer_sizes': (60,),
 'max_iter': 1500,
 'solver': 'lbfgs'}

y_true, y_pred = y_test, clf_nsc.predict(X_test)
print(classification_report(y_true, y_pred))

      precision    recall   f1-score   support
avg / total       0.98       0.98       0.98      540
```

As we can see, the results provided by the nescience library are slightly better in terms of train/test evaluations. However, what it is important is the library has opted for a smaller model (one layer of 60 neurons instead of one layer of 100 neurons) that provides a better result by increasing the maximum number of iterations (from 1000 to 1500). Nescience always select the smallest model that provides the best possible accuracy that does not overfit the training data.

■

17.7 Auto Machine Classification

The nescience library also includes a module for auto-machine learning (both for classification and regression problems). The auto-machine learning module returns the model, from a collection of families of models, that provides the smalles nescience. For each family of models, the class perform a greedy search over the hyperparameters required for each family. In Appendix XX is described the detail for each family of models.

Next example shows how to apply the automachine learning tools.

■ **Example 17.17** In this example we are going to see how to apply the nescience library to find the best model that describes the digits dataset.

```
from sklearn.datasets import load_digits
from sklearn.model_selection import train_test_split

from Nescience.Nescience import AutoClassifier

(X, y) = load_digits(return_X_y=True)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1)

model = AutoClassifier()
model.fit(X_train, y_train)

model.score(X_test, y_test)
[] 0.9622222222222222
```

If we write type(model.model) we will see that the library has selected a linear support vector machine as the best model for this dataset.

■ TODO: Compare with other automl tools.

17.7.1 Surfeit of Algorithms

Decision Trees

For the representation of a tree as a string we use the following template:

```
def tree{[attrs]}:
    if [attr] <= [thresh]:
        return [label] || [subtree]
    else:
        return [label] || [subtree]
```

Where `[attrs]` is the list of attributes used, and only those used in the model,⁴ `[attr]` is a single attribute represented by the letter X followed by a number (e.g. `X1`), `[thresh]` is the threshold used for the split, `[label]` is one of the valid labels from the set \mathcal{G} , and `|| [subtree]` means that the `return` statement can be replaced by another level of `[if - else]` conditions. We could have used a much shorter description of trees by replacing word tokens with symbols, e.g., by the ternary conditional operators `? and :` used in modern programming languages, or by dropping the `return` statement. This would produce shorter trees, but the complexity of the models would remain the same, up to an additive constant that does not depend on the model itself. Since the harmonic mean compares relative values instead of absolute ones, this additive constant can be safely ignored.

17.8 Auto Machine Regression

17.9 Time Series

■ TODO: Introduce this section

17.9.1 Automiscoding, Crossmiscoding and Partial Automiscoding

In this section we are going to study the application of the concept of miscoding to a time series and a delayed version of itself, or to a delayed version of a second time series.

Automiscoding is the application of miscoding to a time series and a delayed version of itself, as a function of this delay. Automiscoding is intended to estimate up to what extend previous observations of the time series can explain (or can be used to forecast) future observations. In this sense, automiscoding has a similar objective than autocorrelation in classical time series analysis (see Section 7.2.6).

Definition 17.9.1 Let $\{\mathbf{x}_t\}$ be a time series composed by n samples. We define lag k *regular automiscoding* of $\{\mathbf{x}_t\}$ as $\hat{\mu}(\mathbf{x}_{x_{k+1}, x_{k+2}, \dots, x_n}, \mathbf{x}_{x_0, x_1, \dots, x_{(n-k)}})$. We define in the same way the concepts of *adjusted automiscoding* and *partial automiscoding*.

On the contrary of what happened with the concept of autocorrelation, automiscoding is defined for all time series, including time series with a trend. Moreover, automiscoding is interpretable even in case of having a trending time series, for example, for the identification of seasonal components without requiring to apply a decomposition.

■ **Example 17.18** In this example we are going to study the presence of cycles in the number of passengers of a US airline. In Figure 17.13 is depicted a time series of monthly passengers from 1949 to 1960 (AirPassenger dataset, see References section bellow). As we can observe, there is

⁴If the dataset contains many attributes, listing all of them when dealing with very short models would make the length of the model's header greater than the length of the body.

a clear cycle that repeats every twelve months. We can apply the concept of auto-miscoding to validate analytically that this is the case.

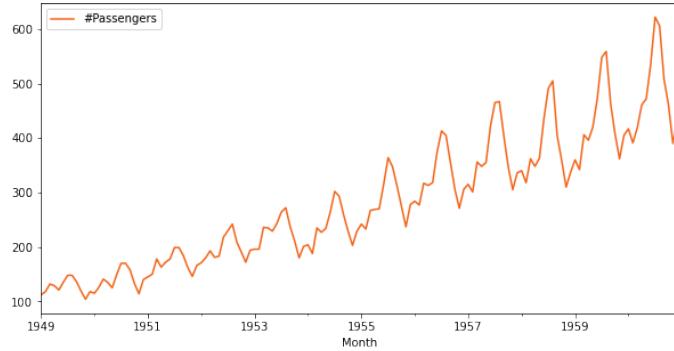


Figure 17.13: Air Passengers.

```
from nescience.timeseries import TimeSeries

data = pd.read_csv("data/AirPassengers.csv", index_col=["Month"], parse_dates=True)
X = np.array(data["#Passengers"]).reshape(-1, 1)

ts = TimeSeries(auto=False)
ts.fit(data)
mscd = ts.auto_miscoding(max_lag=36)
```

As we can see in Figure 17.14 there is a peak on the value of adjusted automiscoding every twelve months (the distance between both time series is minimal when we the lag is a multiple of a year).

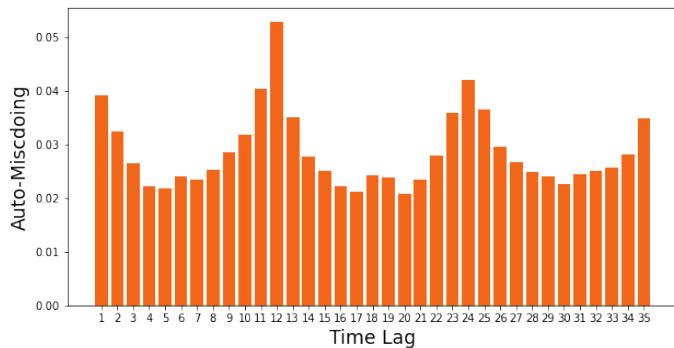


Figure 17.14: Auto-miscoding of Air Passengers.

Crossmiscoding computes the inter-relation between a time series and a lagged version of a second time series. The objective is to detect if the first time series has a temporal predictive power over the second.

Definition 17.9.2 Let $\{\mathbf{x}_t\}$ and $\{\mathbf{y}_t\}$ be two time series composed by n samples each. We define lag k regular crossmiscoding of $\{\mathbf{x}_t\}$ and $\{\mathbf{y}_t\}$ as $\hat{\mu}(\mathbf{x}_{x_{k+1}, x_{k+2}, \dots, x_n}, \mathbf{y}_{x_0, x_1, \dots, x_{(n-k)}})$. We define in the same way the concepts of adjusted crossmiscoding and partial crossmiscoding.

Example 17.19 We are interested to determine if it possible to predict the energy consumption of the appliances of a house. The data set to study (appliances energy prediction dataset, see

References below) is composed by the temperature and humidity conditions measured in the different rooms of the house every ten minutes, and the energy consumption of the appliances. The dataset also includes some information about the current weather, from a nearby weather station.

For every feature we will compute the optimal lag at which the features has the best prediction capabilities:

```
from fastautoml.fastautoml import Miscoding

X = pd.read_csv("../data/energydata_complete.csv", parse_dates=["date"], index_col="date")
y = X["Appliances"]
X = X.drop(["Appliances", "lights"], axis=1)

mCoding = Miscoding()
mCoding.fit(X, y)

best_lag = list()
for i in np.arange(X.shape[1]):
    mscd = mCoding.cross_misCoding(attribute1=i, min_lag=1, max_lag=30)
    best_lag.append(np.where(mscd == np.max(mscd))[0][0] + 1)
```

In Figure 17.15 we can see a plot of the results. As we can observe, in general, for the in-house measurements we should use small lag values. However, in case of the weather conditions, bigger lags provide better results.

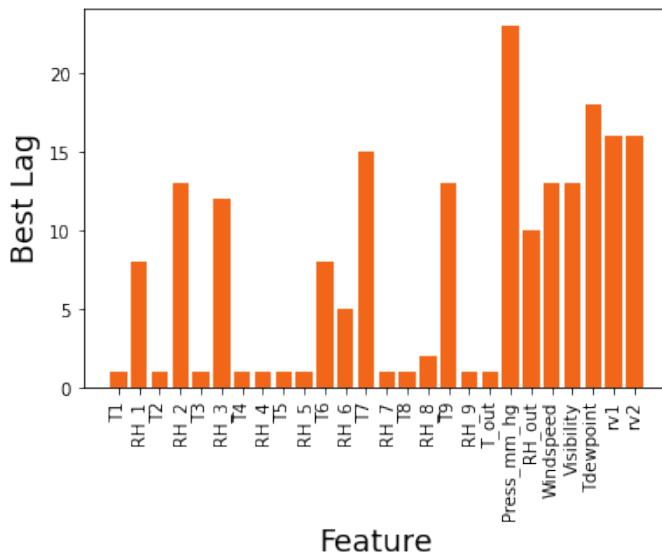


Figure 17.15: Cross MisCoding Lag

R The approximation to the concept of miscoding introduced in this chapter estimates the quality of individual features as predictors of a target value, and the quality of the training dataset as a whole. However, the current approximation does not take into account the existing redundancy among the features themselves. For example, it might happen that two features x_i and x_j have very low miscoding with respect to the target variable y , but at the same time they are redundant, in the sense that they contain almost the same information. It is still an open question how to extend the concept of miscoding to take into account feature redundancy, in such a way that it is close to the theoretical definition, it can be computed efficiently, and it does not require a huge number of samples.

17.9.2 Auto Time Series

TODO: Introduce the auto-time series models, and clearly state what it can be expect from such models (short story: nothing)

TODO: Introduce structural time series models, the state space representation, and the Kalman filter

structure approach [...] different unobserved components or building blocks responsible for the dynamics of the series such as trend, seasonal, cycle, and the effects of explanatory and intervention variables are identified separately before being put together in a state space model.

State space methods originated in the

eld of control engineering, starting with the groundbreaking paper of Kalman (1960). They were initially (and still are) deployed for the purpose of accurately tracking the position and velocity of moving objects such as ships, airplanes, missiles, and rockets [...] these ideas could well be applied to time series analysis generally as well.

In a state space analysis the time series observations are assumed to depend linearly on a state vector that is unobserved and is generated by a stochastically time-varying process (a dynamic system). The observations are further assumed to be subject to measurement error that is independent of the state vector. The state vector can be estimated or identified once a sufficient set of observations becomes available.

Definition 17.9.3 A linear Gaussian state space model for the multivariate time series $\mathbf{y} = \mathbf{y}_1, \dots, \mathbf{y}_n$, where each observation is a p dimensional vector $\mathbf{y}_t = \{y_{t1}, \dots, y_{tp}\}$, is given by

$$\mathbf{y}_t = \mathbf{Z}_t \boldsymbol{\alpha}_t + \mathbf{d}_t + \boldsymbol{\varepsilon}_t \quad \boldsymbol{\varepsilon}_t \sim N(0, \mathbf{H}_t) \quad (17.1)$$

called *space? observation or measurement equation*, and

$$\boldsymbol{\alpha}_t = \mathbf{T}_t \boldsymbol{\alpha}_{t-1} + \mathbf{c}_t + \mathbf{R}_t \boldsymbol{\eta}_t \quad \boldsymbol{\eta}_t \sim N(0, \mathbf{Q}_t) \quad (17.2)$$

called *state or transition equation*, where the individual summands correspond to:

- \mathbf{y}_t observed or measured values,
- \mathbf{Z}_t design matrix,
- $\boldsymbol{\alpha}_t$ unobserved state,
- \mathbf{d}_t observation intercept,
- $\boldsymbol{\varepsilon}_t$ observational disturbance,
- \mathbf{H}_t observational disturbance covariance matrix,
- \mathbf{T}_t transition matrix,
- \mathbf{c}_t state intercept,
- \mathbf{R}_t selection matrix,
- $\boldsymbol{\eta}_t$ state disturbance, and
- \mathbf{Q}_t state disturbance covariance matrix

The $p \times m$ matrix Z_t links the observation vector y_t with the unobservable state vector α_t and may consist of regression variables. The $m \times m$ transition matrix T_t determines the dynamic evolution fo the state vector [...] the observation and state disturbances ε_t and η_t are assumed to be serially independent and independent of each other at all time points [...] matrix R_t is an $m \times r$ selection matrix with $r < m$.

The initial state vector α_1 is assumed to be generated as $\alpha_1 \sim NID(a_1, P_1)$, independen of the observation and estate disturbances ε_t and η_t . Mean a_1 and variance P_1 can be treated as given

konw.

Talk about initialization?

For example, if the time series \mathbf{y} is unidimensional and the state space model is time invariant (only \mathbf{y}_t and α_t depends on t , being the rest of the summands constant), a model with m unobserved states will be given by

$$\mathbf{y}_t = [z_1 \dots z_m] \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} + d_t + \boldsymbol{\varepsilon}_t$$

Some of the most common time series models are particular cases of the state-space model (see Example XX).

■ **Example 17.20**

TODO: Explain the Kalman filter

The *Kalman filter* is a recursive formula that provides an optimal estimate for the unknown state in a state space model. At each time step t , the Kalman filter computes the predicted state conditional to the observations up to time $t - 1$.

Kalman filter can be use for filtering, prediction and smoothing. Here we are only interested in prediction [...] forward pass [...] recursive formulas

Definition 17.9.4

$$\begin{aligned} \mathbf{a}_{t+1} &= \mathbf{T}_t \mathbf{a}_t + \mathbf{K}_t \mathbf{v}_t \\ \mathbf{K}_t &= \mathbf{T}_t \mathbf{P}_t \mathbf{Z}_t^T \mathbf{F}_t^1 \\ \mathbf{v}_t &= \mathbf{y}_t - \mathbf{Z}_t \mathbf{a}_t \end{aligned} \tag{17.3}$$

called *prediction equations*, and

$$\begin{aligned} \mathbf{F}_t &= \mathbf{Z}_t \mathbf{P}_t \mathbf{Z}_t^T + \mathbf{H}_t \\ \mathbf{L}_t &= \mathbf{T}_t - \mathbf{K}_t \mathbf{Z}_t \\ \mathbf{P}_{t+1} &= \mathbf{T}_t \mathbf{P}_t \mathbf{L}_t^T + \mathbf{R}_t \mathbf{Q}_t \mathbf{R}_t^T \end{aligned} \tag{17.4}$$

called *updating equations*.

■ **Example 17.21** **TODO:** Provide an example where we can see how the Kalman filter integrates the predicted probability distribution and observed probability distribution to make a prediction ■

TODO: Examplain the space search algorithm

■ **Example 17.22** **TODO:** Provide an example of auto-time series ■

17.10 Anomaly Detection

As we have seen in Section XXX, the main problem in the area of anomalies detection is that we do not have a precise mathematical definiton of what an anomaly is. Given a dataset, in this book we propose to equate the concept of abnormal samples with that of incompressible samples, and study its consequences. The essence of this chapter is that learning is about finding regularities in a dataset, and finding regularities is what data compression is about. We have also seen that the best model is the one that minimizes the sum of the length of the model plus the length of the data given the model. This optimal model divides our dataset into two disjoint subsets, the compressible part, and the incompressible part. It is the former in which we are interested in this section, since being

incompressible means that they cannot be explained by the model, that is, they are model-based anomalies given the best possible model.

Fix the following definition.

Definition 17.10.1 Let X be a dataset composed by p features and n samples, y the target variable, and M a model such that the nescience $N(X, M)$ is minimal. Let $\hat{y} = M(X)$ be the predictions made by the model M over the vectors of X . We define the *anomaly subset* of X , denoted by \mathcal{A}_M^X , to the set of X such that $y \neq \hat{y}$.

The `nescience.anomalies` class allow us to identify the anomaly subset, that is, the collection of samples that do not match the regularity patterns found in the rest of the dataset. In Example 17.23 we will see how to apply this class to indentify houses with abnormal low prices, and to explain why they are cheaper.

■ **Example 17.23** In this example we will use the Boston House Price dataset provided by the scikit-learn toolkit. The dataset contains 13 predictive features (both, numeric and categorical) measuring different characteristics of the houses, such as number of rooms, age, etc., and the target is the median value of owner-occupied homes. The dataset is composed by five hundred samples. We are interested in to identify those house that have an abnormally low price, that is, houses that given their characteristics (number of rooms, size, etc) should have a higher price.

```
from sklearn.datasets import load_boston

data = load_boston()
X = data.data
y = data.target
```

We have to train a "knowledge model", that is, find the best model that explains the target variable given the predictors, without overfitting. By default, the `anomalies` class uses the AutoML capabilities provided by the `nescience` library.

```
from nescience.anomalies import Anomalies

model = Anomalies(X_type="mixed", y_type="numeric")
model.fit(X, y)
```

Finally, we have to select those samples for which the actual price is smaller than the one predicted by the model.

```
anomalies = model.get_anomalies("smaller")
X.shape, anomalies.shape
((506, 13), (25,))
```

As we can see, there are 25 houses with abnormally low price. ■

The `anomalies` class also allow us to classify the identified anomalies according to the characteristics they share.

Let's see which are the best attributes that describe those abnormal houses.

```
model.get_classes(n_dims=1, an_type="smaller", filter_balancedness=True,
                  filter_redundancy=False, filter_repeated_attrs=False)

Attribute1 Attribute2 Inertia      N Class 0 N Class 1 Ratio
2        None    154.953975 9       16      0.36
4        None     0.090593 6       19      0.24
5        None    5.002437 17       8      0.68
7        None   20.352117 6       19      0.24
8        None   77.611111 18       7      0.72
9        None   41737.11111 7      18      0.28
10       None   26.577436 13       12      0.52
12       None   430.294828 18       7      0.72
```

According to the inertia, the best attribute that allow us to classify the anomalies is the number 4 (nitric oxides concentration). This attribute divides the abnormal houses into two clusters of size 6 and 19. Let's see how the house's price is affected by this dimension.

```
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

lr = LinearRegression()
lr.fit(X[:,4].reshape(-1, 1), y)

plt.scatter(X[:,4].reshape(-1, 1), y)
plt.plot(X[:,4], lr.intercept_ + lr.coef_ * X[:,4], color="red")
plt.xlabel(data.feature_names[4])
plt.ylabel("Price")
```

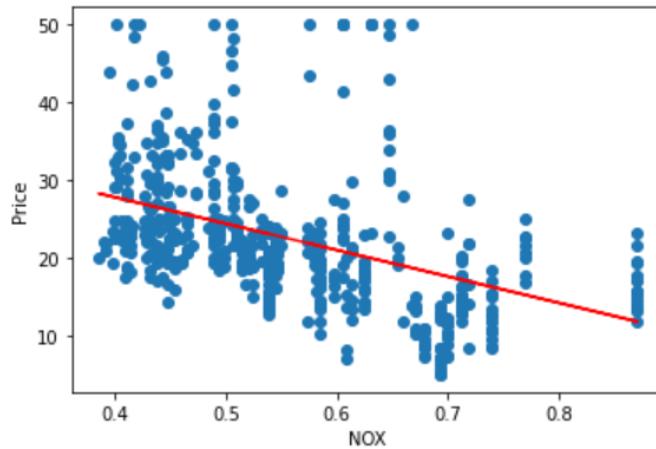


Figure 17.16: Price as a function of NOX.

The regression line suggest that the price of the houses is smaller in those areas with higher levels of nitric oxides concentration. Let's see how the anomalies are classified according to this dimension.

```
class0, class1 = model.get_class_points(attribute1=4, attribute2=None, an_type="smaller")

plt.hist(class0)
plt.hist(class1)
plt.ylabel("Count")
plt.xlabel(data.feature_names[4])
plt.show()
```

The analysis suggests that six of the houses have an abnormally low price because they are in an area with high levels of contamination. We can repeat the same analysis with the other attributes, although they have a higher inertia, so the class separation is not that evident. Please mind that there could be more than one reason why the price of a house is abnormally low.

17.11 Decision Trees

In the last sections we have seen how to use the concepts of miscoding, inaccuracy, surfeit and nescience to evaluate the quality of datasets and models, and to automatically select a family of models and search over its hyperparameters to find the best possible description of a topic. In particular, we have studied in detail the family of binary decision trees. The procedure used in the fastautoml library with trees was a mix between a classical approach (a CART algorithm combined with a cost-complexity pruning), and an evaluation of candidate trees using the minimum

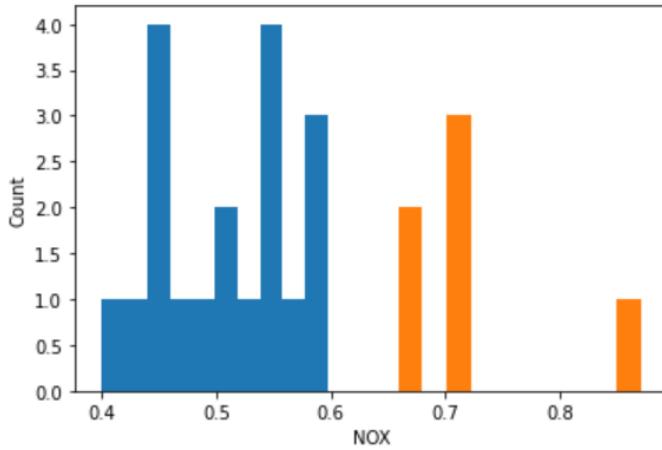


Figure 17.17: Histogram of anomalies NOX.

nescience principle. In this section we are going to see a new algorithm to derive optimal trees, both for classification and regression problems, that is entirely based on the theory of nescience. The new algorithm, by design, avoids the overfitting of the training dataset without losing accuracy, it does not require the optimization of hyperparameters, thus significantly reducing the training time, and it produces much smaller and more shallow trees than traditional algorithms, facilitating the interpretability of the results.

17.11.1 Algorithm Description

The following pseudocode shows the proposed algorithm to build a decision tree given a training dataset (\mathbf{X}, \mathbf{y}) . The procedure is based on a breadth first traversal of trees combined with a greedy approach. It requires a function called `BESTSPLIT()` that returns the best split of a given subset of the data into two subsets; and a second function, called `NESCIENCE()` that provides an estimation of the nescience of the current tree. The algorithm is based on two nested loops: the external `while` loop keeps a list of the candidate nodes to grow, whereas the internal `for` loop finds the best node to grow the tree. The latter operation requires to check all possible growing options and select the one that minimizes the nescience. The exit point of the algorithm is when there are no more branches to grow. We keep track of the best nescience achieved during the building process and return the associated tree.

```

def BUILD_TREE(data)

    nodesList <- list()
    tree <- BESTSPLIT(data)
    bestNescience <- NESCIENCE(tree)
    nodesList.append(tree)

    while not nodesList.empty()

        nescience <- bestNescience
        bestNode <- None
        childNode <- None

        for i <- 1, nodesList.length()

            node <- nodesList[i]

            node.child <- BESTSPLIT(node.ldata)
            tmp <- NESCIENCE(tree)

```

```

        if tmp < nescience
            nescience <- tmp
            bestNode <- i

        node.left <- None

        if nescience < bestNescience
            node <- nodeList[bestNode]
            bestNescience <- nescience
            nodesList.append(node.left)

        if not node.left.empty() and not node.right.empty()
            nodesList.remove(bestNode)

    return tree

```

The main difference of our algorithm from other decision tree building algorithms is in the way the tree is evaluated. Instead of using only accuracy as most of the algorithms do, in addition, we take into account the complexity of the tree (surfeit) to avoid overfitting, and the quality of the subset of data used during the training process (miscoding).

Nescience

The calculation of the nescience implemented in the algorithm is based on a Euclidean distance utility function (see Section 17.6), because that one was the one that produced the best results in the tests we have performed. For the computation of miscoding and inaccuracy, we use the same techniques that the one used in the `fastautoml` library, described in Section 17.3 and Section 17.4 respectively. For the implementation of surfeit, we use the same template to describe trees that was used in the `DecisionTreeClassifier` of the `AutoClassifier` class, and that was described in Section 17.7.1. The only difference is that we also allow equalities in the nodes (if `[attr] = [thresh]`), something not supported by the `DecisionTreeClassifier` algorithm of the `scikit-learn` library.

The generic problem of the instability of inaccuracy due to very short models, also applies to this algorithm (see Section 17.5), and the particular problem of the algorithms to build decision trees, in which the best local split might not be that one that minimizes the error (see Section ??) is also relevant in this case.

The concept of nescience is used in two different ways in our algorithm. For every iteration of the `for` loop we have to decide which one of the candidate branches of the tree we should develop. Recall that the order in which we develop the branches is important, since it might happens that one branch does not get developed because that would mean increase the surfeit without a sufficiently large decrease of the inaccuracy. The second place is at the end of the `while` loop, where we keep track of the nescience of the different building steps, to decide at the end of the algorithm with which tree we return.

We treat regression problems as classification problems in which we discretizes the continuous target variable y into n intervals given the number of samples, and using a uniform discretization (see Section). Once the target variable has been discretized, we train a regular classification tree.

Splitting Criteria

Given a subset $\mathbf{Q} \subseteq \mathbf{X}$ we have to find a split for \mathbf{Q} such that the values of y are grouped together. Recall that a split is a pair $\theta = (j, w)$, where $1 \leq j \leq p$ is a feature index and w is the partition point (see Section 7.2.5). A split divides the set \mathbf{Q} into two disjoint subsets \mathbf{Q}_l and $\mathbf{Q}_r = \mathbf{Q} \setminus \mathbf{Q}_l$. In case of a continuous variable we have that $\mathbf{Q}_l = \{\mathbf{x}_i \in \mathbf{Q} : x_{ij} \leq w\}$, and if the feature is categorical we define $\mathbf{Q}_l = \{\mathbf{x}_i \in \mathbf{Q} : x_{ij} = w\}$ ⁵.

⁵Ideally, for the categorical case, instead of a single feature w we should search over all the elements of the power set of the set of features $\mathcal{P}\{1, 2, \dots, p\}$. Unfortunately, that would imply to check 2^p cases, something that is time-expensive from the computational point of view.

In Section 7.2.5 we saw that a common splitting criteria used in practice is to minimize the weighted entropy \tilde{H} of the subsets \mathbf{Q}_l and \mathbf{Q}_r , that is, to find a split that it is minimal $\theta^* = \arg \min_{\theta} \tilde{H}(\mathbf{Q}, \theta)$. More explicitly, if \mathbf{y} is a target vector taking values from a set of k labels $\mathcal{G} = \{g_1, \dots, g_k\}$ (either because is a categorical target or a continuous target that has been discretized into k intervals), and denoting the subsets of \mathbf{y} as $\mathbf{y}^l = \{y_i : \mathbf{x}_i \in \mathbf{Q}_l\}$ and $\mathbf{y}^r = \{y_i : \mathbf{x}_i \in \mathbf{Q}_r\}$, and n_l and n_r are the number of elements of \mathbf{y}^l and \mathbf{y}^r respectively, we have that

$$\begin{aligned}\tilde{H}(\mathbf{Q}, \theta) &= \frac{n_l}{n} \left(- \sum_{i=1}^k \frac{\sum_{j=1}^{n_l} I(y_j^l = g_i)}{n_l} \log_2 \frac{\sum_{j=1}^{n_l} I(y_j^l = g_i)}{n_l} \right) \\ &\quad + \frac{n_r}{n} \left(- \sum_{i=1}^k \frac{\sum_{j=1}^{n_r} I(y_j^r = g_i)}{n_r} \log_2 \frac{\sum_{j=1}^{n_r} I(y_j^r = g_i)}{n_r} \right)\end{aligned}\quad (17.5)$$

In our nescience based decision tree algorithm, the splitting criteria is to minimize the total length of encoding the subsets \mathbf{Q}_l and \mathbf{Q}_r using optimal codes. We have to find the optimal split $\theta^* = \arg \min_{\theta} \hat{K}_C(\mathbf{Q} | \theta) = \arg \min_{\theta} \{\hat{K}_{C_l}(\mathbf{Q}_l) + \hat{K}_{C_r}(\mathbf{Q}_r)\}$ where C_l and C_r are the optimal codes given the relative frequencies of the observed values of \mathbf{y}^l and \mathbf{y}^r respectively. The quantity $\hat{K}_C(\mathbf{Q} | \theta)$ is computed as:

$$\hat{K}_C(\mathbf{Q} | \theta) = \hat{K}_{C_l}(\mathbf{Q}_l) + \hat{K}_{C_r}(\mathbf{Q}_r) = - \sum_{i=1}^k \log_2 \frac{\sum_{j=1}^{n_l} I(y_j^l = g_i)}{n_l} - \sum_{i=1}^k \log_2 \frac{\sum_{j=1}^{n_r} I(y_j^r = g_i)}{n_r}$$

In this particular case (if we use as compression algorithm a code with optimal lengths, and continuous variables have been discretized) it turns out that both expressions are equivalent given the following relation:

$$\tilde{H}(\mathbf{Q}, \theta) = \frac{1}{n} \hat{K}_C(\mathbf{Q} | \theta)$$

We prefer to talk of encoding length instead of weighted entropy because it has an easier interpretation in the context of the theory of nescience.



Strictly speaking, if we want to implement a decision trees search algorithm fully compliant with the minimum nescience principle, instead of using a total length encoding as splitting criteria, we should have computed the nescience at each split and select that one that makes it minimal. However, early experiments have shown that at local level it works better to group the values of y than to reduce the nescience. Further research is required to confirm and explain this point.

Practical Implementation

In the web page that accompanies this book⁶ we provide an open-source implementation of our algorithm in Python. Our software can be used together with other machine learning tools from the `scikit-learn` library, since we adhere to their API guidelines. For example, our algorithm can be used as part of an ensemble of classifiers, like the `BaggingClassifier` meta-estimator, or the results of the classification could be cross-validated with tools like `cross_val_score`. As an example, to provide a model for the breast cancer dataset, we could do something like the following:

```
from NescienceDecisionTree import NescienceDecisionTreeClassifier
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()

model = NescienceDecisionTreeClassifier()
model.fit(data.data, data.target)
```

⁶<http://www.mathematicsunknown.com>

```
print("Score: ", model.score(data.data, data.target))
```

17.11.2 Algorithm Evaluation

In this section we are going to evaluate our new algorithm, and compare its performance against the well-known algorithm CART. CART, *Classification and Regression Trees*, is the de-facto standard algorithm used in the machine learning industry for the derivation of decision trees. For this particular experiment we have used the CART implementation provided by scikit-learn.

Figure 17.18 shows a synthetic dataset consisting of 1000 random points lying on a two dimensional plane, where all the points with an X_1 attribute less than 50 are colored blue, and the rest as red. We have artificially introduced a red point, simulating a measurement error, in the blue area. The black lines correspond to the decisions performed by CART. Since the CART algorithm will not stop until all the points have been properly classified, we have to specify an expected count condition to limit the number of splits. The figure correspond to the tree generated by CART setting the `min_samples_leaf` hyperparameter to 5.

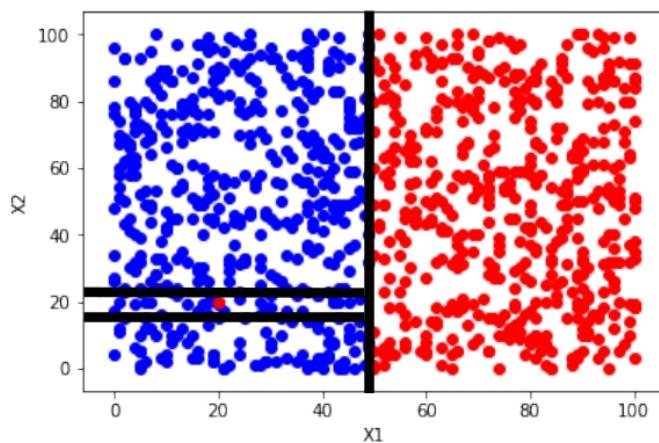


Figure 17.18: Synthetic dataset with CART algorithm splits.

The tree obtained by applying our algorithm to the dataset of Figure 17.18 can be seen in Figure 17.19. The nescience based algorithm does not try to model the error point, since the gain due to an increment in the accuracy does not compensate the surfeit introduced in the model. Recall that the algorithm stops when the total nescience of the tree, based on the measures of miscoding, inaccuracy and surfeit, does not decrease when adding new nodes to the tree. Our algorithm presents a lower sensitivity to the errors found in datasets, at least if the number of errors is small compared with the number of valid points.

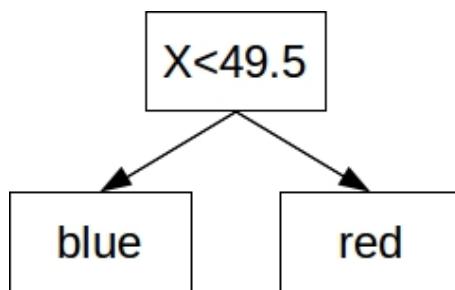


Figure 17.19: Decision tree obtained by the nescience algorithm.

Our second experiment, again with synthetic dataset, is depicted in Figure 17.20. There, we

create two isotropic Gaussian blobs that partially overlap. We start with a standard deviation of 2.5 for each cluster, so they are easy to separate, and we increase the standard deviation in increments of 0.01, until we reach 4.5, which causes significant overlaps. For each value of the standard deviation, we run the experiment 100 times and we compute the average accuracy for the two algorithms using different datasets for training (70% of the data) and testing (30% of the data). The results of this experiment are shown in Figure 17.21.

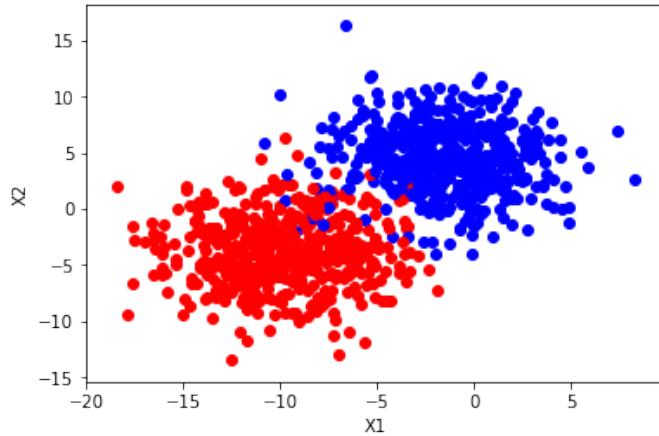


Figure 17.20: Isotropic Gaussian Blobs.

As we can see, the performance of both algorithms, in terms of accuracy, is similar. However we should note that the hyperparameter `minimum_leaf_size` of the CART algorithm has been optimized to achieve the best accuracy. For this particular experiment, the best value was achieved with a minimum leaf size of 26 points. By definition, given the fact that CART has one degree of freedom more than the nescience algorithm, it should produce better accuracy; something that it is not observed (both algorithms have a mean accuracy of 0.87).

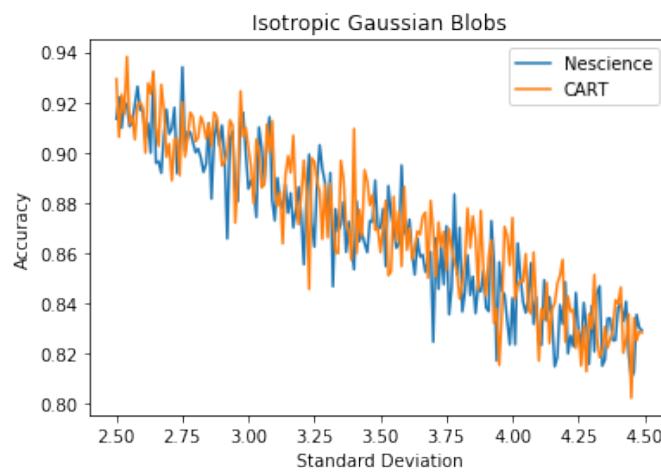


Figure 17.21: Accuracy of Isotropic Gaussian Blobs.

For each iteration of the experiment, we have also computed the average number of nodes, including internal and leaf nodes, required by the models to properly classify the clouds in the dataset. The results of this measurement are show in Figure 17.22. Our algorithm requires an average of 4 nodes compared to 23 nodes for the CART algorithm. Moreover, our algorithm is

more stable than CART, in the sense that it produces models of similar complexity when it gets similar input datasets (a standard deviation of 0.31 compared to 3.77 for CART).

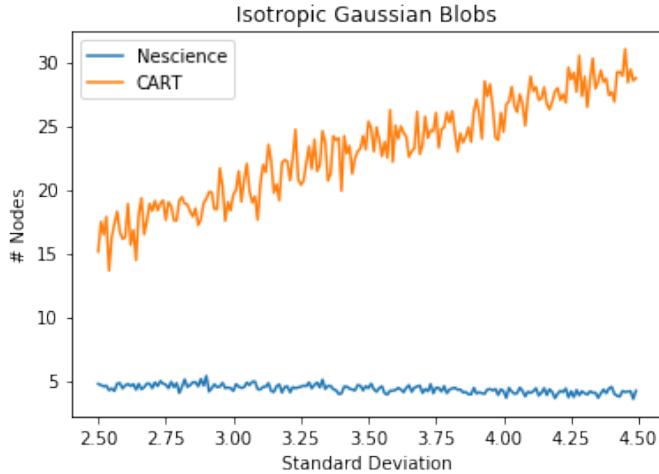


Figure 17.22: Number of Nodes.

In Figure 17.23 we show the maximum depth of the tree, defined as the longest path from the root of the tree to any of its leaves. The maximum depth of the tree is a good measure of the average time it will require for the model to provide a classification. The nescience algorithm has an average depth of 1.6 nodes, whereas the average depth yielded by the CART algorithm is 4.8 nodes.

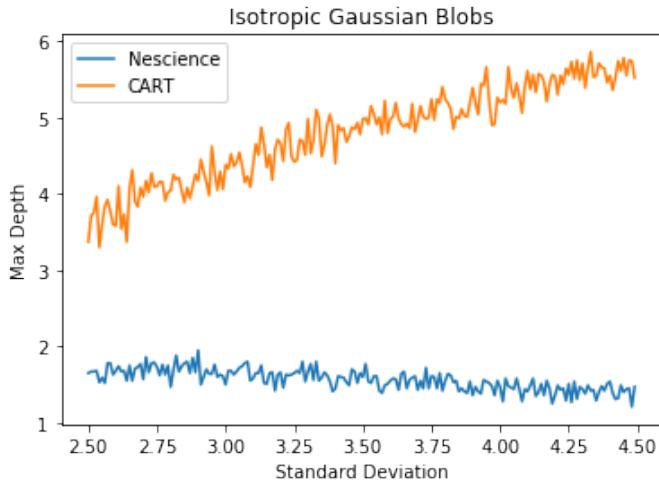


Figure 17.23: Maximum depth of the model.

The last part of the evaluation consists in comparing the performance of our algorithm and CART with a collection real datasets. More specifically, we have selected 12 well known datasets from the UCI Machine Learning Repository. The selected datasets are: diagnosis of breast cancer (`cancer`), optical recognition of handwritten digits (`digits`), predicting protein localization sites in gram-negative bacteria (`yeast`), classification of NASA space shuttle data (`shuttle`), classification of blocks in web pages (`page`), segmentation of outdoor images (`image`), predicting the age of abalones from physical measurements (`abalone`), predicting the quality of red and white variants of Portuguese wine (`wine`), filter spam emails (`spam`), wall-following robot navigation (`wall`),

classification of land use based on Landsat satellite images (`landsat`), and distinguishing signals from background noise in the MAGIC gamma telescope images (`magic`). For each dataset, we have repeated the experiment 100 times, by randomly selecting the training (70%) and testing (30%) subsets at each iteration.

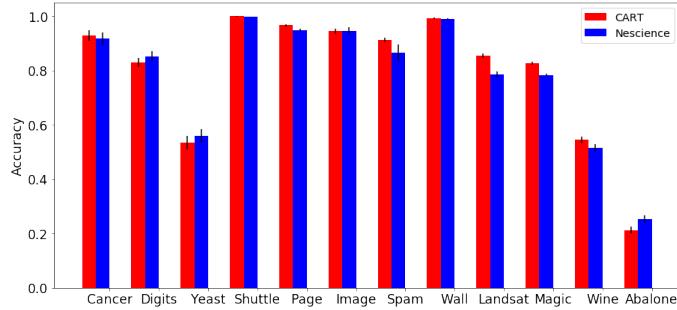


Figure 17.24: Maximum depth of the model.

In Figure 17.24 we compare the accuracy of the resulting models obtained by applying the CART algorithm and the nescience algorithm to the above datasets. In 4 of the 12 datasets, our algorithm provides better accuracy than CART. In the remaining 8 cases, the accuracy is, on average, less than 1% smaller.

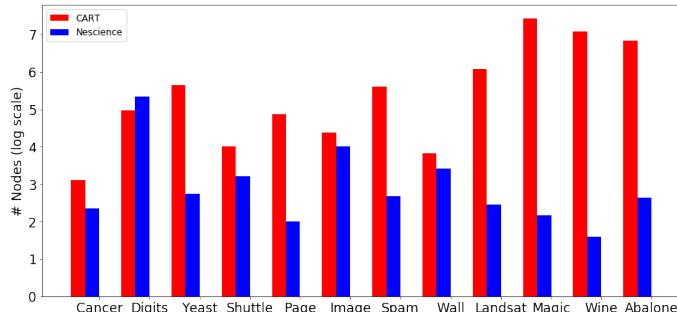


Figure 17.25: Maximum depth of the model.

In Figure 17.25 it is shown a comparison of the total number nodes (internal nodes plus leaf nodes) of the resulting models. Only for one of the datasets (`digits`), our model produces a slightly more complex tree than those generated by CART. In the rest of the cases, the number of nodes in the trees generated by the nescience algorithm have between two and three orders of magnitude fewer nodes (in this figure the y axis is in logarithmic scale).

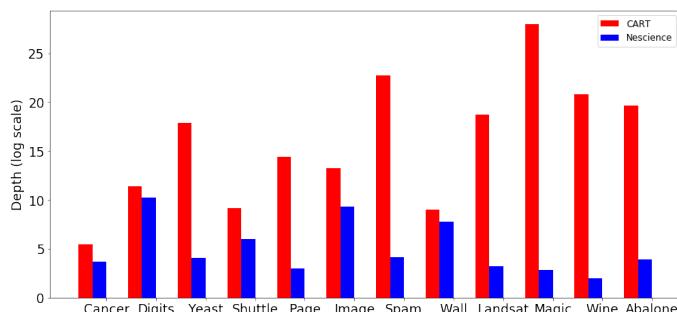


Figure 17.26: Maximum depth of the model.

Finally, in Figure 17.26 we provide a comparison of the depth of the tree of the resulting models. Our algorithm always yields a shallower tree than the CART algorithm.

We would like to mention that the nescience algorithm is highly robust with respect to the compressor selected or the nescience function implemented. In Table 17.11.2, we have applied the nescience algorithm to the datasets described above, and evaluate different alternatives for the definition of the nescience function $N(X, M)$: arithmetic mean $(\mu(M, D) + \tau(X, M) + \sigma(M, D))/3$, geometric mean $(\mu(M, D) + \tau(X, M) + \sigma(M, D))^{1/3}$, harmonic mean $3/(\mu(M, D) + \tau(X, M) + \sigma(M, D)) - 1$, Euclidean distance $(\mu(M, D) + \tau(X, M) + \sigma(M, D))^{1/2}$, sum $\mu(M, D) + \tau(X, M) + \sigma(M, D)$, and product $\mu(M, D) + \tau(X, M) + \sigma(M, D)$. The table shows limited difference between the different functions.

	Euclid	Arithmetic	Geometric	Product	Addition	Harmonic
Accuracy	0.758	0.784	0.803	0.803	0.784	0.81
Stdev	0.051	0.041	0.033	0.033	0.041	0.038

Table 17.2: Comparison of nescience functions

Similarly, Table 17.11.2 shows the performance of our algorithm when using the LZMA, zlib, and bz2 compressors. We observe that all of them yield similar performance. The above results suggest that the performance our algorithm is independent of the specific choice made for either implementation aspect.

	bz2	lzma	zlib
Accuracy	0.813	0.804	0.81
Stdev	0.03	0.045	0.038

Table 17.3: Comparison of compressors

We emphasize that the CART algorithm requires to optimize a configuration hyperparameter in order to obtain good results, whereas the algorithm proposed in this book does not require from this optimization.

Shallower trees means faster forecasting times when the models used in production, since the number of `if-else` conditions to be evaluated is smaller. Moreover, smaller trees makes easier to interpret the results by human analysts, and much shorter training times, something very relevant in case of training ensembles of trees, like random forest or boosted trees (although the use of ensembles of models is highly discouraged by the theory of nescience, given their high surfeit).

17.12 Algebraic Model Selection

As it was the case for the definition of nescience based on the encyclopedic description of research topics, the nescience of structured datasets can be used to evaluate alternative descriptions of research topics (mathematical models), and to identify how far these descriptions are from an ideal perfect knowledge. This evaluation could be used to identify those topics which require further research. Moreover, the same methodology could be applied to collections of datasets to identify our current knowledge of research areas (collections of topics).

If we combine the concept of nescience of a model, with our concepts of relevance and applicability of research topics, we could apply our methodology for the assisted discovery of interesting questions to collections of datasets; a very useful methodology now that big datasets are becoming widely available.

In order to evaluate the methodology developed, we are going to apply it to a particular research topic: *Multipath Wave Propagation and Fading*. The problem at hand is to understand the effect of a propagation environment on a radio signal, such as the one used by wireless devices. The signals reaching the receiving antenna could follow multiple paths, due to atmospheric reflection and refraction, and reflection from water and objects such as buildings. The effects of these multiple wave paths include constructive and destructive interference (fading), and phase shifting of the original signal, resulting a highly complex received signal (see Figure 17.27).

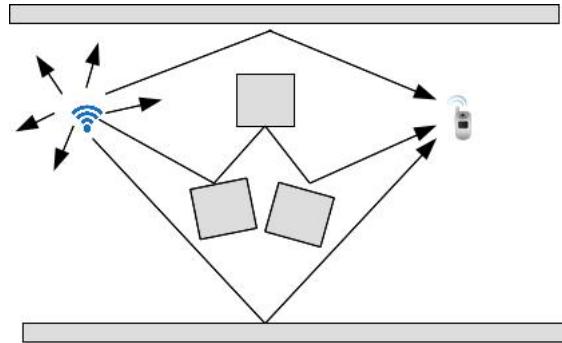


Figure 17.27: Multipath Signal Propagation

In many circumstances, it is too complicated to describe all reflection, diffraction and scattering processes that determine the different paths the signal will follow. Rather, it is preferable to describe the probability (stochastic model) that the received signal attains a certain value. We are interested in to analyze how well these stochastic models (our current knowledge) are able to describe what happen in reality.

The *Rayleigh fading model* assumes that the magnitude of a signal will vary randomly, or fade, according to a Rayleigh distribution (the radial component of the sum of two uncorrelated Gaussian random variables). The Rayleigh probability density function of the power signal is given by:

$$P_\sigma(x) = \frac{1}{\sigma} \exp\left[-\frac{x}{\sigma}\right]$$

where σ is the mean of the received signals. Rayleigh fading is viewed as a reasonable model for the effect of heavily built-up urban environments, when there is no dominant propagation along a line of sight between the transmitter and receiver.

The Rice or *Rician distribution* describes the power of the received signal when the target consists in many small scatterers of approximately equal strength, plus one dominant scatterer whose individual received signal equals all that of all the small scatterers combined (there is a dominant line of sight). The probability density function of the power of the received signal is given by:

$$P(x) = \frac{1}{\bar{\sigma}} (1 + a^2) \exp\left[-a^2 - \frac{x}{\bar{\sigma}} (1 + a^2)\right] I_0\left[2a\sqrt{(1 + a^2) \frac{x}{\bar{\sigma}}}\right]$$

where $\bar{\sigma}$ is the mean of the received signals, and it is equal to $\bar{\sigma} = (1 + a^2) \bar{\sigma}_R$, being $a^2 \bar{\sigma}_R$ the power of the dominant scatterer, and I_0 is the modified zeroth order Bessel function of the first kind.

An experiment (see Figure 17.28) was set up to collect a real dataset to analyze. The experiment was run on a $135m^2$ office full of obstacles (interacting objects). The transmitter was an Odroid C1 Linux computer with a Ralink RT5370 USB Wifi adapter. The receiver was a (fixed in space)

Motorla Moto G mobile phone. Data was collected using the Kismet⁷ platform (an 802.11 layer2 wireless network detector, sniffer, and intrusion detection system), with some ad hoc, home made, software extensions, mostly for data aggregation. A total of 3,177 samples (power level measured in dBm) were collected during one hour experiment.

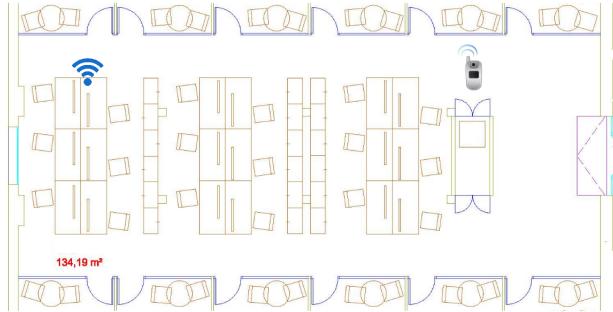


Figure 17.28: Experimental Set Up

Next table summarizes the results of applying the three considered models (uniform, Rayleigh and Rice) and the optimal encoding using a Huffman code:

Model	LDM	Nescience
Uniform	17,351	1.30
Rayleigh	13,229	0.75
Rice	11,118	0.47
Huffman	7,541	-

Table 17.4: Nescience of Models

The uniform model, that is, assuming zero knowledge about the topic covered by the dataset, has a nescience of 1.30. This value is a kind of upper level for the nescience associated with that particular topic and dataset; any model with a higher nescience should be classified as zero knowledge model. If we introduce the knowledge that in a environment with multiple obstacles the signal propagation can be described as a Gaussian process (Rayleigh distribution), we are able to decrease our nescience to 0.75, that is, there were a 43% improvement in our understanding of the topic. If we add the knowledge that there is usually a strongly dominant signal seen at the receiver caused by a line of sight between the antenna and the mobile phone (Rician distribution), the nescience decreases to 0.47, and so, we have achieved an additional 23% gain in our understanding. Given that numbers we can conclude that the Rayleigh model increases our knowledge with respect to the uniform model, and that the Rice model does so with respect to Rayleigh. However, the nescience of this last model is 0.47. That means that there still patterns in the dataset that are not explained by the Rice model, or what it is equivalent according to our methodology, there is still some knowledge to discover and learn.

The methodology has been applied to a dataset gathered in a single experiment under a controlled environment, since the goal of this Chapter was to provide a methodology to quantify the nescience of structured datasets, not to evaluate models for signal propagation and fading. In order to conclude that, in general, the Rice model is an improvement over Rayleigh, a more realistic experiment is required, with multiple datasets gathered in real environments.

⁷<https://www.kismetwireless.net/index.shtml>

17.13 The Analysis of the Incompressible

As we have said in Chapter chap:Introduction, one of the reasons to understand how things work is to understand the cause-effect relation in systems. We are interested in this cause/effect relation in two ways. That is, if we want to see an effect in a system, we want to understand which causes trigger that effect. Also, and perhaps more interesting, if we have observed an (probably undesired effect) in a system we would like to discover what has caused that effect, so we can fix it, and revert the normal situation.

We could use the theory of nescience to model, and modify, those uncommon effect, by means of training a model and looking at the incompressible part of the data.

A model \mathcal{M} for a dataset $\mathcal{D} = (X, y)$ is a compressed version of that dataset, since the length of the dataset given the model $l(\mathcal{D} | \mathcal{M})$ is smaller than the length of the original dataset $l(\mathcal{D})$. The model \mathcal{M} is composed by the regularities found in the dataset (subject to the algorithm used and the families of models considered). What is left, $\mathcal{D} | \mathcal{M}$ is the incompressible part of the dataset, that is, those samples that have no regularity at all, or present a regularity that requires a description longer than the length of the raw data.

In this section we are going to show the practical applications of analyzing what is left, that is, the incompressible samples of a dataset. An element that is incompressible represents a very unlikely, or uncommon, situation of the entity being studied. A incompressible element does not necessarily mean a problem, since if a problem is sufficiently common, it can be compressed. An incompressible element is something that cannot be explained given the normal behaviour of the system. Of course, all of this is assuming that our dataset has no errors.

Once we have found a model that has the lowest possible nescience for a dataset, we could separate those elements that have not been compressed, denoted by \mathbf{XX} , and fit a second model. We could argue that it does not make any sense to model the incompressible part, since, it is incompressible. However, the incompressible part is incompressible with respect to the original entity under study, that is, the global system. And in this new case, we are studying a different entity, namely, the uncommon parts of an entity. It might happen that we can find regularities in this new entity.

References

A insightful description of the differences between explanation models and predictive models, how these models are used in different scientific disciplines, and what are the implications for the process of statistical modeling can be found in [[shmueli2010explain](#)].

The application of the minimum description length principle to the identification of optimal decision trees have been proposed in [[quinlan1989inferring](#)], further refined and clarified in [[wallace1993coding](#)]; however the coding method proposed by those authors is different from the one used in this book.

In our web page can be found an implementation of the decision tree following the guidelines defined in [].

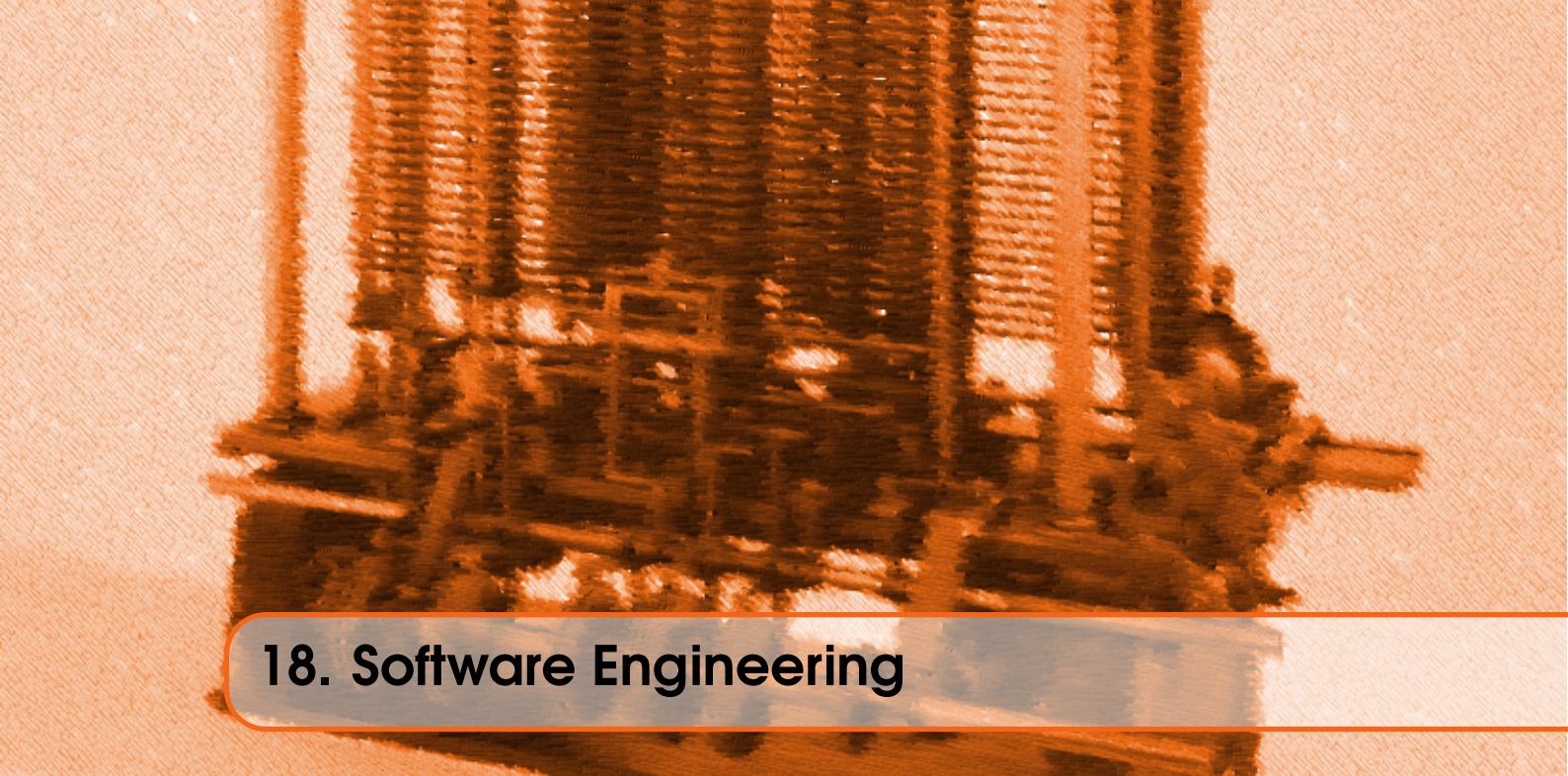
The Minimum Description Length [[grunwald2007minimum](#)] and the Minimum Message Length [[wallace2005statistical](#)] techniques have been applied to the problem of inferring decision trees in [[quinlan1989inferring](#)], later on clarified and extended in [[wallace1993coding](#)], in [[mehta1995mdl](#)] as a technique for pruning, and in [[rastogi1998public](#)], among others. Although the underlining concepts behind the cost function proposed in this chapter are the same (namely, that learning is equivalent to the capability to compress), our approach is very different from the ones described in these works.

An excellent survey of the available discretization methods can be found in [[garcia2013survey](#)]; in the paper the authors also propose a taxonomy to classify existing methods based in their

properties and they conduct an extensive comparative experimental study. The proportional discretization method used to compute miscoding is introduced in [yang2009discretization], where there is also a theoretical justification of why this method reduces the bias and the variance of the discretized variable.

TODO: Find the original reference of the AirPassengers dataset.

TODO: Find the original reference of the Appliance energy consumption dataset. <https://archive.ics.uci.edu/ml/datasets>



18. Software Engineering

TODO: Change chapter image

TODO: Rewrite this introduction

Our final example of how to apply the theory of nescience in practice comes from the area of software engineering. The goal of this example is to show how we can apply the concept of nescience to a problem where there are no descriptions. As I have said, what it is a description depends on the particular application at hand.

In the particular case of software engineering we want to solve two related problems. The first one is to quantitatively measure how mature is a particular software platform. The second one is how to discover new software tests, that have not been considered by the programmers, in that way we could increase the quality of the software.

The relevance will be based on software modules and data flow, and the nescience will be based on the number of lines of code for these modules. The rationale is that very well understood tasks are usually coded in libraries, meanwhile not very well understood problems are usually resolved ad-hoc.

Of course, the above statement is just conjectures that need more research to be confirmed, perhaps analyzing hundred of software projects, and perhaps, performing some experiments. The goal of this chapter is to show how the theory of nescience can be applied to other areas than scientific research, not to provide a quantitative measure of software quality.

18.1 Redundancy of Software

The concept of nescience applied to software engineering allows us to measure how immature is a particular software application or platform. As it was the case for scientific research topics, if we can compress a source code, probably we do not fully understand how to solve the problem at hand. The rationale is that immature software usually contains a lot of duplicated, redundant, code; meanwhile mature software it is usually composed of generic functions and libraries that provide near optimal solutions to common problems. As a consequence, if a code contains redundant elements there must exist a better way to solve the problem.

In this section we are going to study and compare the nescience of three software platforms in the area of database management systems: SQLite, PostgreSQL and Cassandra. The first two, SQLite and PostgreSQL, are relational database management systems based on SQL, and the third one, Cassandra, is a noSQL database (knowledge of relational databases or SQL is not required to understand this chapter). SQLite and PostgreSQL platforms are based on the C programming language, meanwhile Cassandra is written in Java. The three platforms are publicly available open source software, so the reader can repeat the experiments himself.

SQLite

SQLite¹, according to the description from its web home page, *is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine*. Unlike other relational databases, SQLite is not based on a client-server model, instead it is composed by a highly compact library (usually less than 500 KiB) that reads and writes directly to ordinary disk files. In fact, in SQLite a complete database with its multiple tables, indices, triggers, and views, is contained in a single disk file.

Table 18.1 provides a compilation of the results of the analysis of a selection of versions of SQLite along 15 years of continuous development (first release was published in August 2000). The columns of the table represent: the version number, the number of source code lines, the size (number of bytes) of the source code, and the size (number of bytes) of the compressed version of the source code (using the Linux tool gzip). The final column contains the nescience associated to each version.

The analysis was performed in a Linux-based machine. The command issued to compute the number of source lines of each version was:

```
# find SQLite-1_0/src -name '*.[chly]' -exec cat {} \; > total-1.0
```

Figure 18.1 depicts the evolution of the nescience of the SQLite platform along the different versions analyzed. As it can be observed, in general, the nescience decrease for each new release: from a nescience of 3.23 for the first, initial release, to a nescience of 2.86 in the latest published version. This behavior of nescience in SQLite is the expected behavior of a software platform that has been designed with a highly targeted goal in mind. Each new release of SQLite is focused in doing better what the software already does, instead of adding a lot of new functionality. Given the evolution of the nescience, we could say that SQLite is a software where for every new release we know better how it manage small restational databases.

PostgreSQL

PostgreSQL² is a full-featured open source object-relational database system, with a strong emphasis in reliability, data integrity, and correctness. PostgreSQL provides advanced database management capabilities, like tablespaces, data replication, fault tolerance, hot backups, and many others. PostgreSQL evolved from another database, the Ingres project, at the University of California, Berkeley. The team released version 1 of the new database to a small number of users in 1989, version 2 with a re-written rules system in 1990, and version 3 in 1991. After releasing version 4.2 in 1994 the project ended. All those initial releases of PostgreSQL were not included in the analysis because they are not open source. In 1996, the project was renamed to its current name PostgreSQL to reflect its support for the language SQL, and moved to an open source MIT-style license, which enabled other developers to modify and extend the source code. The first open source PostgreSQL release was version 6.0 from 1997. Currently, PostgreSQL is developed and maintained by the *PostgreSQL Global Development Group*, that comprises companies and individual contributors.

¹<http://www.sqlite.org>

²<http://www.postgresql.org>

Version	Lines	Size	Compressed	Nescience
1.0	11,668	363,025	84,317	3.31
2.0	20,629	623,058	151,382	3.12
2.1	22,389	681,026	165,631	3.11
2.2	22,829	695,475	169,191	3.11
2.3	24,246	745,092	181,251	3.11
2.4	26,574	822,638	201,335	3.09
2.5	30,110	943,328	231,555	3.07
2.6	31,154	977,407	238,490	3.10
2.7	31,667	995,136	243,210	3.09
2.8	35,357	1,117,115	274,299	3.07
3.0	49,197	1,547,363	390,292	2.96
3.1	54,646	1,722,646	437,335	2.94
3.2	55,940	1,763,913	449,046	2.93
3.3	65,498	2,070,135	529,485	2.91
3.4	82,288	2,606,260	666,321	2.91
3.5	86,369	2,737,174	700,228	2.91
3.6	97,864	3,093,778	790,732	2.91
3.7	126,341	4,158,239	1,080,647	2.85
3.8	148,506	4,923,732	1,274,720	2.86
3.9	166,744	5,583,265	1,446,884	2.86
3.10	168,451	5,636,155	1,458,698	2.86

Table 18.1: Nescience of SQLite

Table 18.2 provides a compilation of the results of the analysis of a selection of versions of PostgreSQL along 25 years of development. The columns in table have the same meaning than in Table 18.1. The command issued to compute the number of source lines was:

```
# find postgresql-1_0/src -name '.*.[chyl]' -exec cat {} \; > total-1.0
```

Figure 18.2 depicts the evolution of the nescience of the PostgreSQL database engine along the different versions that have been published. As it can be observed, in general, the nescience increase for each new release: from a nescience of 3.30 for the first version analyzed to a nescience of 3.56 in the latest published version. This behavior of nescience in PostgreSQL is the expected behavior for a software platform that has the goal of provide as much new functionality as possible in every new release. Even if the developers have improved the already existing code, the immaturity of the new code produces an increment of the nescience of the platform. Given the evolution of the nescience, we could say that PostgreSQL is a software where for every new release it does more things but we know less about how it does them.

Cassandra

Apache Cassandra³ is an open source distributed database management system focused in scalability and high availability of data. Cassandra can provide near linear data scalability and fault-tolerance on commodity hardware. Cassandra data model is based on column indexes instead of the normalized tables of relational (SQL based) models. Cassandra was initially developed at Facebook to power their search feature, and it was released as an open source project on Google code in 2008. In 2009, it became an Apache Incubator project, and in 2010 it became a top-level project.

³<http://cassandra.apache.org>

Version	Lines	Size	Compressed	Nescience
1.09	178,538	4,857,018	1,128,931	3.30
1.09	178,976	4,869,670	1,132,454	3.30
6.0	187,950	5,190,331	1,219,104	3.26
6.1	200,488	5,541,679	1,287,908	3.30
6.2	222,602	5,559,815	1,298,418	3.28
6.3.2	260,809	6,715,908	1,531,409	3.39
6.4.2	297,918	7,711,626	1,759,091	3.38
6.5	330,540	8,933,109	1,959,557	3.56
7.0	376,445	10,133,053	2,298,580	3.41
7.1	409,314	11,156,575	2,557,159	3.36
7.2	443,499	12,189,722	2,804,470	3.35
7.3	461,091	13,095,834	2,935,691	3.46
7.4	525,512	14,938,894	3,371,228	3.43
8.0	586,198	16,735,624	3,797,590	3.41
8.1	628,324	18,149,522	4,105,712	3.42
8.2	676,591	19,608,764	4,409,352	3.45
8.3	780,706	22,969,821	5,056,367	3.54
8.4	862,793	25,819,421	5,619,089	3.59
9.0	927,850	27,947,706	6,016,580	3.65
9.1	992,814	29,922,940	6,464,460	3.63
9.2	1,054,939	31,889,273	6,921,880	3.61
9.3	1,090,891	32,824,041	7,146,213	3.59
9.4	1,149,347	34,698,419	7,576,636	3.58
9.5	1,247,679	37,834,424	8,300,979	3.56

Table 18.2: Nescience of PostgreSQL

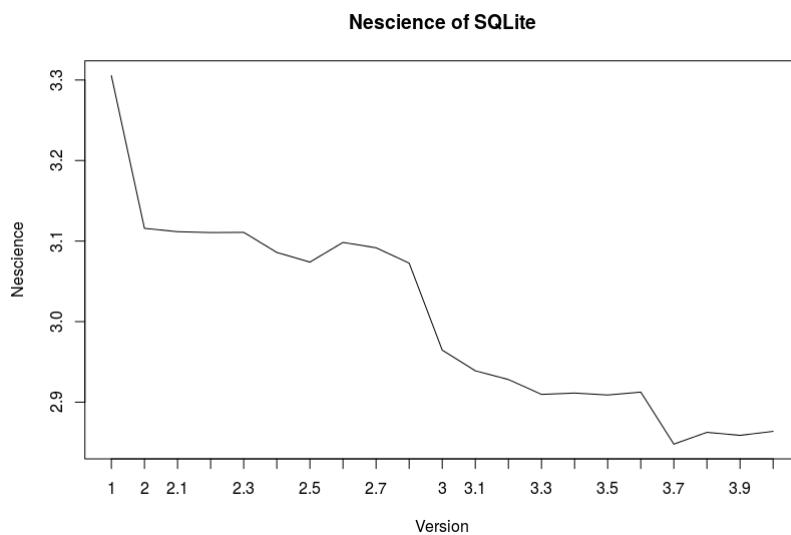


Figure 18.1: Nescience of SQLite

Table 18.3 provides a compilation of the results of the analysis of a selection of version of Cassandra along 7 years of development. The columns in table have the same meaning than in Table 18.1 and Table 18.2, so the nescience of the three projects can be compared. The command issued to compute the number of source lines was:

```
# find cassandra-0.3.0/src/ -name "*.java" -exec cat {} \; > total-0.3.0
```

Figure 18.3 depicts the evolution of the nescience of the Cassandra platform along the different versions that have been analyzed. As it can be observed, the nescience presents a first period in which it clearly decreased for every release, and a second period where it increases again. A possible explanation of this behavior could be that Cassandra was a highly immature project at its initial releases (a nescience of 4.89) and during these first versions the developers improved the quality of the source code, and during the second period the developers focused more in to add new functionality.

18.2 Quality Assurance

Table 18.4 shows the ten most relevant functions of SQLite. The functions call graph has been computed with the aid of the cflow utility. cflow is a program that analyzes source files written in the C programming language and outputs the call graph between various functions. **TODO: explain**

Table 18.5 shows the ten functions with higher nescience. **TODO: explain**

Finally, Table 18.6 shows the ten most interesting functions from the point of view of testing. **TODO: explain**

18.3 Forex Trading Robots

Open source mql4 robots evaluated with metatrader4 in the EUR/USD exchange over a period of one year (2016), at intervals of five minutes, and with a fixed spread of 2 pips.

Version	Lines	Size	Compressed	Nescience
0.3.0	50,370	1,747,132	296,565	4.89
0.4.2	35,386	1,217,575	214,303	4.68
0.5.1	36,823	1,286,675	231,918	4.55
0.6.13	39,740	1,402,412	251,556	4.57
0.7.10	64,279	2,321,786	419,346	4.54
0.8.10	77,331	2,764,743	503,978	4.49
1.0.12	88,479	3,152,422	575,984	4.47
1.1.12	106,441	3,857,420	712,932	4.41
1.2.19	142,775	5,240,648	942,257	4.56
2.0.16	165,593	6,177,890	1,105,372	4.59
2.1.9	195,424	7,315,659	1,316,190	4.56
2.2.1	212,249	7,928,612	1,399,814	4.66
3.0.0	242,320	9,083,317	1,617,120	4.62

Table 18.3: Nescience of Cassandra

Function	Relevance
sqlite3_free	203
sqlite3_malloc	87
sqlite3_mprintf	69
sqlite3_mutex_leave	59
sqlite3_mutex_enter	57
sqlite3SafetyCheckOk	36
fts3SqlStmt	35
sqlite3_realloc	32
sqlite3_mutex_held	29
sqlite3Fts3GetVarint	20

Table 18.4: Most relevant functions of SQLite

Function	Description	Complexity	Nescience
fts5PorterStep4	3,012	429	6.02
fts5PorterStep2	4,101	600	5.83
sqlite3ErrName	6,424	1,049	5.12
sqlite3ExprIfTrue	4,015	988	3.06
winFullPathname	5,976	1,475	3.05
unixSectorSize	2,795	697	3.01
sqlite3GetToken	5,967	1,490	3.00
jsonParseValue	3,856	968	2.98
sqlite3ExprIfFalse	5,180	1,301	2.98
sqlite3TreeViewExpr	6,789	1,731	2.92

Table 18.5: Highest nescience in SQLite functions

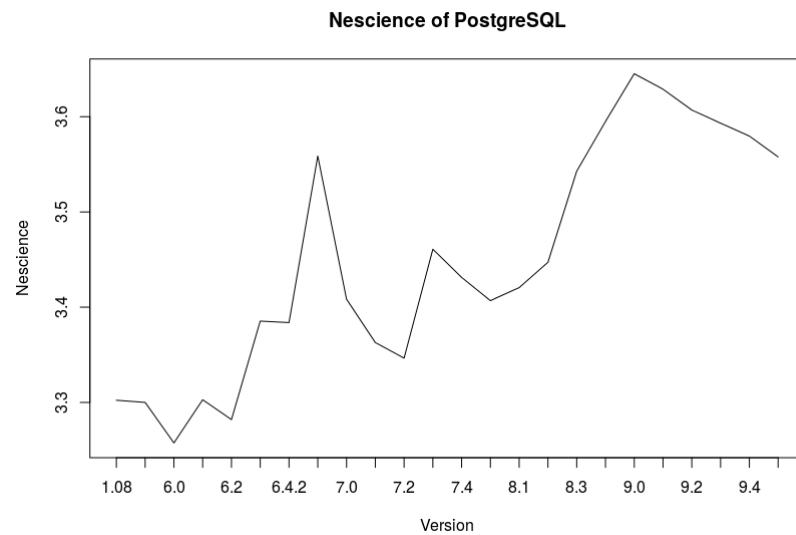


Figure 18.2: Nescience of PostgreSQL

Function	Interestingness
fts3SqlStmt	0.55
sqlite3_free	0.54
sqlite3_initialize	0.47
sqlite3VXPrintf	0.46
fts5CheckTransactionState	0.43
fts5StorageGetStmt	0.43
sqlite3Fts5GetVarint	0.41
sqlite3TreeViewExpr	0.38
fts5DataRead	0.38
sqlite3Fts3SegReaderStep	0.38

Table 18.6: Interestingness of SQLite functions

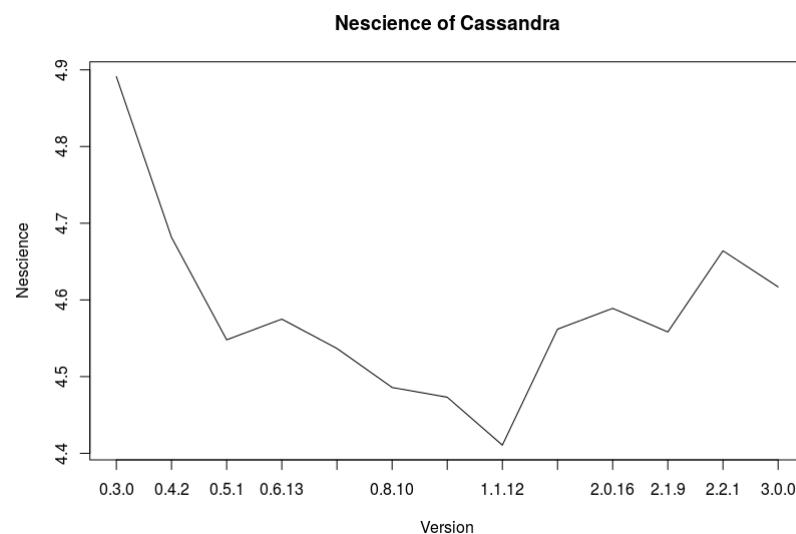
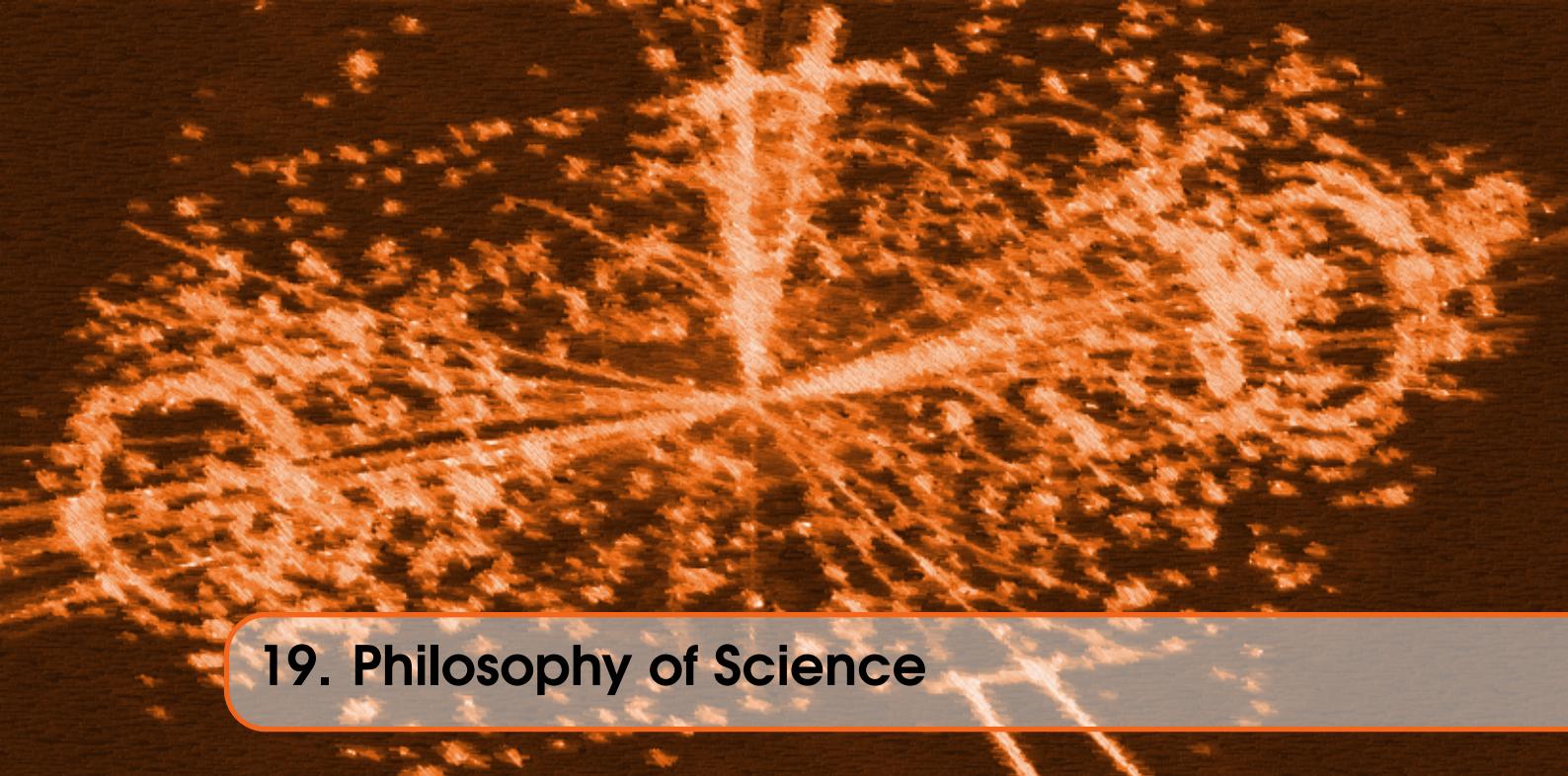


Figure 18.3: Nescience of Cassandra



19. Philosophy of Science

Science may be regarded as the art of data compression.
Li & Vitányi

19.1 Wikipedia, The Free Encyclopedia

the analyses performed will be based on the collection of scientific pages from Wikipedia. As we will see, Wikipedia pages not only contain the encyclopedic coverage of research topics that we need for the theory of nescience, they are also a highly cited reference in the scientific community, and they are very well known by the general public. These two additional characteristics make this collection of articles a highly valuable resource to validate our methodology in practice.

Wikipedia pages are written in the *MediaWiki Markup Language*, which provides a simple notation for formatting text, so that users without knowledge of XHTML or CSS can edit them easily. MediaWiki tags are often misused by editors, therefore it is required to apply several heuristics in order to circumvent those problems. We have used an advanced *Wikipedia Extractor* utility to extract the relevant text of the pages. Also the extractor was instructed to remove all those non relevant elements for our analysis, such as images, tables, references and lists.

19.2 Classification of Research Topics

TODO: Talk about compressors, its behavior depending of size of object and window (buffer) size, and the speed-compression ratio trade off.

The Kolmogorov complexity of a page was estimated using the compressed version of the text. As compressed tool we have used the Unix *gzip* utility, that it is based on a combination of the Lempel-Ziv LZ77 algorithm and Huffman coding. Figure 19.1 shows a plot of nescience for all the topics after normalization.

Table 19.1 contain the ten topics with highest nescience, and the normalized version of this number. The identification of the topics with highest nescience is even more controversial. Some

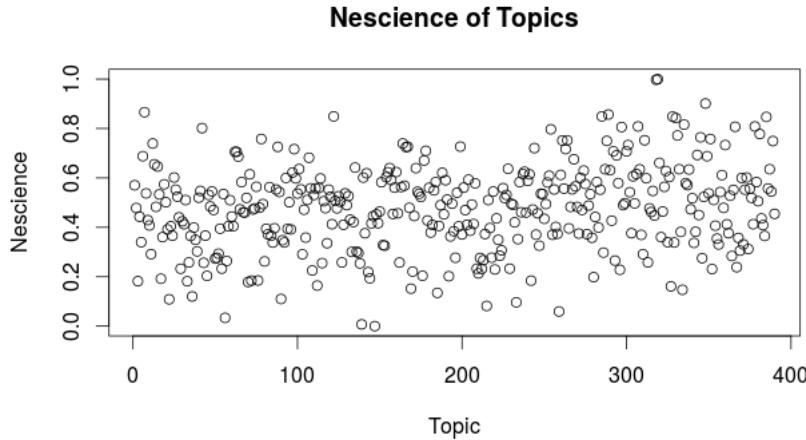


Figure 19.1: Nescience of topics

Topic	Nescience	Norm.
Arithmetical hierarchy	2.28	1.00
Analytical hierarchy	2.28	1.00
Hyperarithmetical theory	2.13	0.90
Kolmogorov struct. funct.	2.08	0.87
Behavior of DEVS	2.06	0.86
UML state machine	2.05	0.85
Computability	2.05	0.85
Kleene's recursion th.	2.05	0.85
Computability theory	2.04	0.84
Computation in the limit	2.00	0.82

Table 19.1: Nescience of topics

authors would argue that the topics listed in the table are the least known topics in the area of theory of computation, like for example *UML state machine* or *Kleene's recursion theorem*. However the list includes very difficult to address topics, *computability* and *computability theory*, and hot research topics like the *Kolmogorov structure function*. It is also remarkable that the list contains three related topics, *analytical hierarchy*, *arithmetical hierarchy* and *hyperarithmetical theory*, suggesting that this is a highly unknown area of knowledge. An important point to mention is that our computed nescience is not directly proportional to the length of the article, and so, the list of the topics with the higher nescience does not mimic the list of the lengthiest articles.

In practice, our definition of nescience is problematic when we work with very short descriptions, since the compressed version of the text could be larger than the uncompressed version, given a negative nescience. This is still an open problem that must be addressed, perhaps by borrowing some techniques from the minimum description length principle.

The maturity of a topic is estimated based on the length of the Wikipedia article (only the text), and the length of the compressed version. Figure 19.2 shows a plot of the maturity of the selected set of topics after the normalization process.

Table 19.2 contains the ten most relevant topics according to its maturity. For each topic it is shown the maturity and the normalized version of this number. Well classified topics, that is, topics that our intuition tell us that are well understood, could include *Read-only right moving Turing*

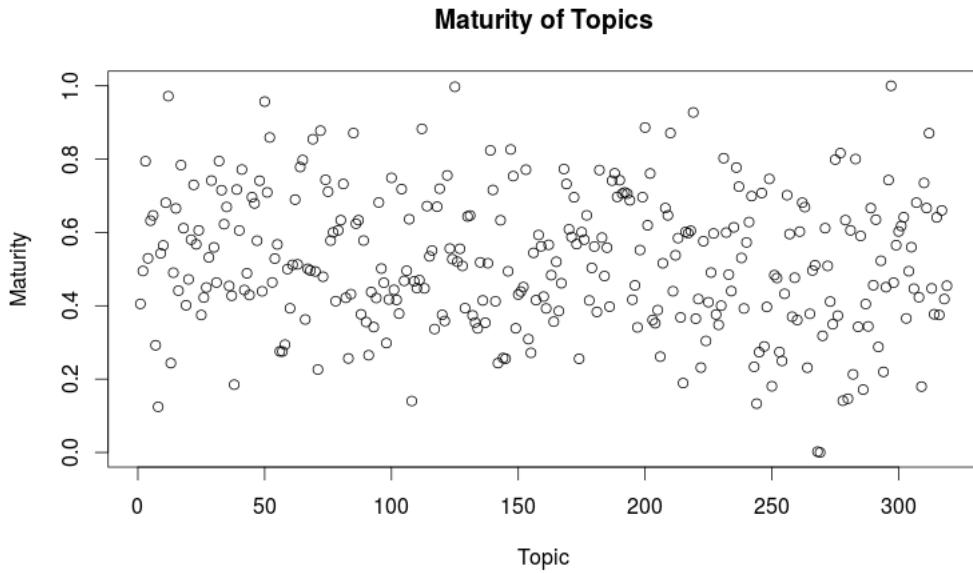


Figure 19.2: Maturity of topics

Topic	Maturity	Norm.
Carry operator	5.34	1.00
Binade	4.54	0.99
Comm. X-Machine	3.01	0.97
PowerDEVS	2.35	0.94
MPIR	2.00	0.92
Constraint automaton	1.84	0.90
RO right moving TM	1.73	0.89
P"	1.71	0.89
Crossing sequence (TM)	1.63	0.88
Microsoft Binary Format	1.53	0.86

Table 19.2: Maturity of topics

machines, *Crossing sequence (Turing machines)*, and perhaps the *P” language*. Other topics that perhaps are misclassified include *communication X-Machine*, *Power DEVS*, *MIPR*, and *constraint automaton*.

The most difficult part of the identification of topics as tools, that is, topics with very high maturity (or very low nescience), is to distinguish when the description of a topic is short because it is well understood (for example, a mathematical theorem), or when it is short because it is a unfinished or poorly written article. Our work is based on the classification of Wikipedia articles as stubs, however, this classification is not very reliable, since many stubs articles are not classified as such (many of the misclassified topics suffer from this problem). How to automatically distinguish between a well-understood topic and a poorly written description is still an open question.

The interest of a topic as a tool measures how likely is that this tool can be applied to other problems. Figure 19.3 shows a plot of the interestingness of the selected set of topics after the normalization process.

Table 19.3 contains the ten most relevant topics according to its interestingness as a source of interesting tools. Out of the ten topics, only two (*ternary numeral system* and *recursion*) appear

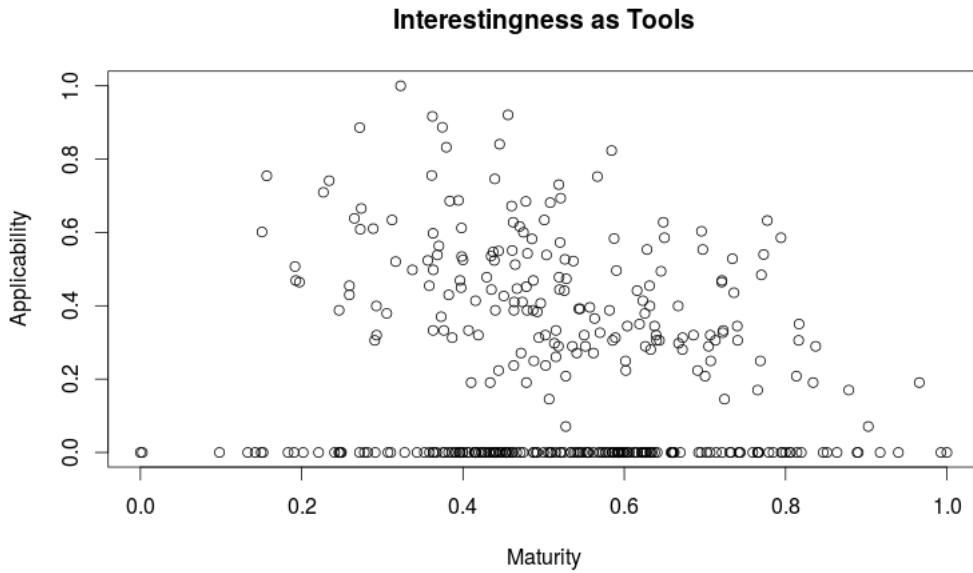


Figure 19.3: Interestingness of Tools

Topic	Interestingness
GNU MPAL	0.49
Ternary numeral system	0.48
IEEE 854-1987	0.47
Arithmetic logic unit	0.43
Recursion	0.42
Barrel shifter	0.42
State space	0.42
Abstract machine	0.41
Computational model	0.39
Arithmetic overflow	0.39

Table 19.3: Interestingness of Tools

in the list of top ten mature topics or top ten applicable topics; the rest of topics are new. In the list we can find topics like the *GNU Multiple Precision Arithmetic Library* and the *standard for radix-independent floating-point arithmetic* (IEEE 854-1987) that are definitely tools, but not in the sense of tool that we are looking for our methodology. Some other topics are not clear that can be considered as tools, like *arithmetic logic unit*, *barrel shifter*, or *arithmetic overflow*. Topics that match or intuitive idea of tool include *recursion*, *state space*, *abstract machine*, and *ternary numeral system*. There are also some topics, like *computational model*, too broad to be considered in a question.

Finally, Figure 19.4 contains a plot of the interestingness of the topics considered as potential interesting problems.

Table 19.4 shows the ten most interesting topics as interesting problems. Topics that fit our intuitive idea of problem, that is, not very well understood concepts with a high relevance, could include *arithmetical theory*, *halting problem*, *floating point*, *quantum computer*, and *computable function*. The topic *recursion* appears both as a tool and as a problem. However in case of tools it refers to the concept of recursion in general, and in case of problems it refers to the implementation

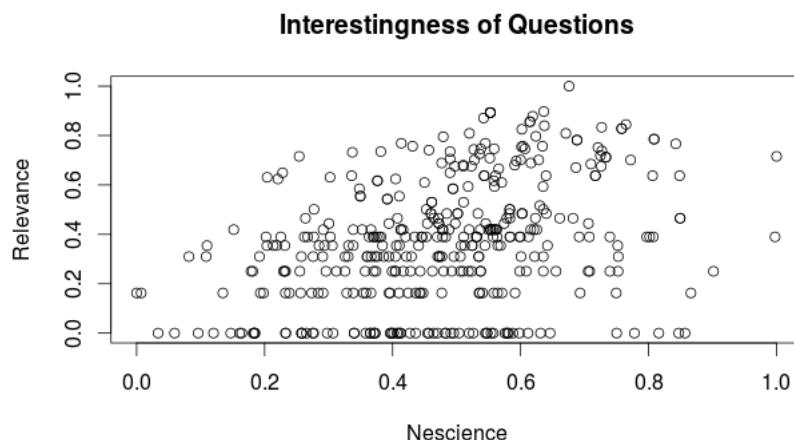


Figure 19.4: Interestingness of Questions

Topic	Interestingness
Arithmetical hierarchy	0.72
Regular expression	0.68
Computability theory	0.65
Halting problem	0.65
Recursion (CS)	0.64
Lambda calculus	0.63
Floating point	0.61
Quantum computer	0.57
Computability	0.55
Computable function	0.55

Table 19.4: Interestingness of Problems

of the concept of recursion in the particular case of computer science. The case of *regular expression*, a topic that intuitively should be classified as a tool and not as a problem, can be explained due to the length of the article in Wikipedia, that provides a detailed description of the language used for regular expressions. This problem rises the question of how to distinguish in Wikipedia between introductory articles and reference articles. Finally, there are topics like *computability theory*, *lambda calculus* and *computability* that are too broad to be analyzed as problems.

R Based on the given definition of "Interestingness of a topic as a tool", we can explore various mathematical properties and concepts derived from this idea. Some possibilities include:

- Normalization: To compare different topics fairly, we can normalize their interestingness values by scaling them to a specific range, e.g., [0, 1]. This normalization can help compare the relative interestingness of various topics.
- Weighted Interestingness: We can introduce weights to the maturity and applicability dimensions to emphasize one over the other depending on the specific context or application. This would allow us to fine-tune the interestingness measure for different scenarios.
- Correlation: Studying the correlation between maturity and applicability could provide insights into how these dimensions are related and possibly reveal trends across different research topics.
- Cluster Analysis: By examining topics in the two-dimensional vector space defined by

Research Area	Applicability	Maturity	Tools
Sociology	1.00×10^{-3}	2.93×10^{-3}	3.09×10^{-3}
Biology	9.20×10^{-4}	4.65×10^{-3}	4.74×10^{-3}
Chemistry	3.11×10^{-3}	5.01×10^{-3}	5.90×10^{-3}
Psychology	1.14×10^{-3}	6.91×10^{-3}	7.00×10^{-3}
Mathematics	9.32×10^{-3}	9.47×10^{-3}	1.32×10^{-2}
Epistemology	1.55×10^{-3}	1.75×10^{-2}	1.76×10^{-2}
Computer_science	9.93×10^{-3}	1.90×10^{-2}	2.15×10^{-2}

Table 19.5: Interestingness of Areas as Tools

maturity and applicability, we can perform cluster analysis to identify groups of topics with similar levels of interestingness. This can help identify areas of research that share characteristics and possibly suggest interdisciplinary research opportunities.

- Rate of Change: Investigating the rate of change of interestingness over time can provide insights into the evolving landscape of a research field. This analysis could reveal emerging topics or those that are becoming less relevant.
- Optimization: Using the interestingness metric, we can explore optimization techniques to find the most interesting topics given certain constraints or within specific domains. This could be useful for research prioritization and resource allocation.

These derived mathematical properties and concepts can provide a deeper understanding of the interestingness of research topics and their potential application as tools for solving problems.

19.3 Classification of Research Areas

In this section we analyze the interests of the different research areas, instead of the interest of individual topics. The areas analyzed are *sociology* (Level 1 *social sciences*, 6,004 topics), *biology* (Level 1 *natural sciences*, 6,989 topics), *computer science* (Level 1 *applied sciences*, 1,331 topics), *epistemology* (Level 1 *cognitive science*, 4,268 topics), *psychology* (Level 1 *behavioral science*, 7,905 topics), *chemistry* (Level 1 *physical sciences*, 11,589 topics), and *mathematics* (Level 1 *formal sciences*, 4,688 topics).

Table 19.5 shows the average applicability and average maturity of each of the selected areas, and the average interestingness of each area as a source of interesting tools. The table largely fits our intuitive idea of which areas are more important as a source of tools: computer science is the area of highest interest, and sociology is the area with less interest. The only strange element is that epistemology appears as a source of very interesting tools, even more interesting, on average, than topics from mathematics (probably because it contains a large ratio of poorly written articles).

Finally, Table 19.6 shows the relevance and nescience of the selected areas, and their interest as a source of interesting problems. Again, the results largely match our intuitive idea of which areas are less understood: sociology is the area with the highest number of interesting problems, and mathematics is the area with the lower number of problems.

19.4 Interesting Research Questions

By combining the elements of Table 19.3 and Table 19.4 we could come up with new interesting ideas of how to apply existing tools to open problems. As it was said above, the goal of the approach described in this article is to identify highly potential interesting applications, but is up to the researcher to decide if certain combination of topics make sense or not, and if they deserve the effort to pursue them. The results of the combination is in Table 19.7.

	Relevance	Nescience	Problems
Mathematics	4.22×10^{-2}	3.51×10^{-1}	3.53×10^{-1}
Computer_science	2.35×10^{-2}	4.43×10^{-1}	4.44×10^{-1}
Chemistry	5.95×10^{-2}	4.66×10^{-1}	4.70×10^{-1}
Biology	3.85×10^{-2}	4.75×10^{-1}	4.77×10^{-1}
Psychology	5.06×10^{-2}	5.28×10^{-1}	5.31×10^{-1}
Epistemology	4.54×10^{-2}	5.30×10^{-1}	5.32×10^{-1}
Sociology	4.21×10^{-2}	5.43×10^{-1}	5.44×10^{-1}

Table 19.6: Interestingness of Areas as Problems

Tool	Problem	Interestingness
Ternary numeral system	Regular expression	1.21
GNU MPAL	Arithmetical hierarchy	1.19
IEEE 854-1987	Arithmetical hierarchy	1.17
Quantum computer	Regular expression	1.17
Ternary numeral system	Arithmetical hierarchy	1.15
Division by zero	Regular expression	1.15
Turing machine	Regular expression	1.14
GNU MPAL	Regular expression	1.13
Ternary numeral system	Halting problem	1.13
Recursion	Halting_problem	1.13

Table 19.7: Interesting Intradisciplinary Questions

Most of the interesting questions identified have very low quality. As it was said before, the problem is that it is very difficult to distinguish (automatically and unsupervised) between a poorly written article from a very well understood topic. In this section we review some on the interesting intradisciplinary questions identified with the aim to clarify what we mean as interesting question and how interesting questions should be interpreted. Some combinations worth examining could be:

- Interesting Question 7: *Can we apply Turing machines to regular expressions?* The answer to this question is yes, since it is a very well known question. Regular expressions are recognized by finite automata, and finite automata can be simulated by Turing machines.
- Interesting Question 10: *Can we apply recursion to the halting problem?* Again the answer is yes, since the proof of the halting problem is based on a machine that calls itself.

Note that both questions have well known answers, and so, we have failed to provide original questions.

The most interesting questions arise when we combine topics from two different disciplines. However, the probability that the identified questions are meaningful is lower than in the case of intradisciplinary analysis.

For the interdisciplinary analysis we have used the collection of pages from the theory of computation already used in the intradisciplinary analysis, and a new collection of topics from the area of bioinformatics. The topics were selected using the Wikipedia category *natural sciences, biology, biological processes*. In total, there were more than 10^5 combinations analyzed. Table 19.8 contains the list of the most relevant intradisciplinary applications.

The set of interdisciplinary questions also suffer from the problem of the stub articles, and so, the quality of the results is low. Some interdisciplinary applications could be:

- Interesting question 1: *Can we apply state space to action potential?* Questions 1, 2, 3, 4

Tool	Problem	Interestingness
State space	Action potential	1.17
Turing machine	Action potential	1.16
Quantum computer	Action potential	1.16
Abstract machine	Action potential	1.14
Computational model	Action potential	1.13
State space	Membrane potential	1.12
State space	Meiosis	1.11
Arithmetic logic unit	Meiosis	1.11
GNU MPAL	Flashbulb memory	1.11
Ternary numeral system	Working memory	1.10

Table 19.8: Interesting Interdisciplinary Questions

and 5, all of them, suggest the same idea, that is, if it is possible to formalize the concept of action potential, in such a way that can be reproduced by a computer.

- Interesting question 7: *Can we apply state space to meiosis?* Question 7 is similar to question 1, and it asks about the possibility of formalize the concept of meiosis using a computer.

19.5 Interesting Research Topics

If we combine the list of highly relevant and not very well understood problems with themselves, it might happen that we come up with a new topic that lies in the new unknown unknown area.

In Figure 19.5 it is shown a plot¹ of the interestingness of the (potential) new topics compared with the interestingness of the topics that generated them.

Table 19.9 contains a list of the top 25 candidates to become new topics topics according to their interestingness. In this analysis we have included all the topics from all the knowledge areas. Most of the questions deal with the concept of intellectual property (*copyright, open access, public domain*, and perhaps, *wiki*), suggesting that this is an area where there are still a lot of things to discover, much more than we are aware of. Perhaps, it could be also a problem of a certain bias of Wikipedia to these, and related, topics. Further investigation is required to clarify this point.

In order to understand how new topics are generated, we have selected the following two examples:

- New topic 17: *Public domain + Earth*. This question rises the issue if the Earth should be considered as a public resource; it touches the very concept of private property. The methodology suggest that this is not a very well understood topic.
- New topic 18: *Public domain + Internet*. Raises the same issue that Question 17, but in this case restricted to Internet and its governance.

Unfortunately, in both cases we fail to provide a well defined, innovative, and previously unseen, research topic.

We could also restrict our search of new topics to a reduced number of knowledge categories. For example, in Table 19.10 contains the ten most interesting new topics corresponding to the already studied areas of *theory of computation* and a new area of *phenomenology* (from Level 2 *philosophy of mind*, and Level 1 *cognitive science*). Given the list of topics contained in the table, we could come up with, for example, the following potential new topics:

- New topic 2: *Turing machine + synesthesia*: this new topic could be about a new kind of Turing machine that incorporates synesthetic properties. These new *synesthetic Turing machines* could be defined as the union of a group of Turing machines that are linked together

¹With the aim to make the figure clear, only a reduced set of the topics is depicted.

Problem	Problem	Interestingness
Public domain	Open access	1.71
Public domain	REST	1.70
Public domain	Wiki	1.70
Open access	REST	1.70
Copyright	Public domain	1.69
Open access	Wiki	1.69
Public domain	QR code	1.69
Copyright	Open access	1.68
Wiki	REST	1.68
Open access	QR code	1.68
Public domain	Transport Layer Security	1.68
Copyright	REST	1.68
QR code	REST	1.67
Open access	Transport Layer Security	1.67
Copyright	Wiki	1.67
Wiki	QR code	1.67
Public domain	Earth	1.67
Public domain	Internet	1.67
REST	Transport Layer Security	1.66
Copyright	QR code	1.66
Earth	Open access	1.66
Internet	Open access	1.66
Public domain	Open source	1.66
Public domain	Web 2.0	1.66
Wiki	Transport Layer Security	1.66

Table 19.9: New Topics

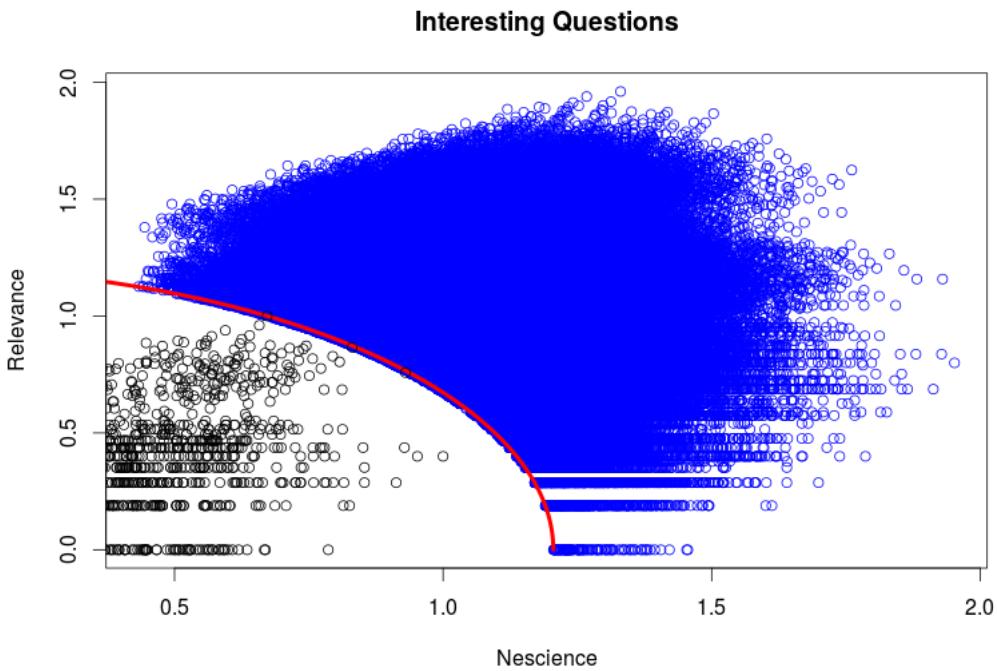


Figure 19.5: Interesting Intradisciplinary Questions

in such a way that when one machines read a symbol from its tape, it produces an automatic change in the state of another machine. The property of synesthesia could be also extended to the case of non-deterministic Turing machines.

- New topic 4: *Kolmogorov Complexity + Self-awareness*: This topic could be interpreted as investigating the minimum complexity required for a computer program to have the capacity of self-awareness.

19.6 Philosophy of Science

Philosophy of science is a branch of philosophy concerned with the foundations and methods of science. The central issues addressed by the philosophers of science are the following:

Question	Question	Interestingness
Kolmogorov complexity	Change blindness	1.24
Turing machine	Synesthesia	1.23
Kolmogorov complexity	Qualia	1.23
Kolmogorov complexity	Self-awareness	1.22
Turing machine	Qualia	1.22
Kolmogorov complexity	Synesthesia	1.21
Turing completeness	Synesthesia	1.20
Turing machine	Self-awareness	1.20
Turing completeness	Qualia	1.20
Turing completeness	Self-awareness	1.18

Table 19.10: Restricted New Topics

- Does science allow us to reach an absolute knowledge?
- What are scientific theories? How do we evaluate competing theories?
- How are theories discovered and evaluated? Is there a universal scientific method?
- What is scientific explanation? What is problem solving? Does science enable progress?
- What is the difference between science and pseudoscience?

Unfortunately, up to today there is no consensus among philosophers about the right answers. Although the theory of nescience has not been explicitly designed to provide a solution to any of these open problems, since it is a theory to be applied in practice, not a framework to understand how we gather knowledge or to explain what science is, we think we could provide a possible interpretation that might bring some light into these fundamental questions. In the next paragraphs we provide a short summary of how these inquiries about science itself could be addressed in the context the theory of nescience, and the rest of the chapter provides the details of the answers that the theory of nescience provides.

Q: Does science allow us to reach an absolute knowledge?

Philosophers of science deal with the problem of how knowledge about our world is gathered through our senses, and if we can trust our perceptions. Also, they address the difficult issue of how knowledge is derived from facts (for example, by means of applying the principle of induction), and if it is sound, from a logic point of view, to make those derivations. Finally, philosophers are interested in how scientific theories are generated based in this knowledge. Any of these problems is covered by the theory of nescience, since we assume that theories (or descriptions in our own terminology) are already known. We do not provide any method to create those theories. What the theory of nescience provides is a set of metrics to quantitatively evaluate, and compare, existing scientific theories.

It might appear that the descriptions in which the theory of nescience is based are truly objective, in the sense that they must be so clear and well stated that even a computer can reconstruct the original topic given its description. Although this point is true, the problem that prevents the theory to provide an absolute knowledge about our world is the way we choose the entities to study, and how we encode as strings those entities. As we have seen (see Chapter 9), the accuracy of our descriptions depend on how good is our encoding of the abstract entities we are studying. Unless the entities are strings themselves, we must assume that our encoding could not be perfect. Moreover, we could be wrong about our assumption that the selected set of entities covers all possible entities of that kind. That is, the set of entities are subject to change as our scientific understanding about them develops. **The same might happen in case of encodings.**

Although the theory of nescience does not say anything about how we can reach an absolute knowledge about an entity, it can tell us if we have reached a perfect description (that we make equal to a perfect knowledge). That is, the theory of nescience can answer the question if we have reached a perfect knowledge about a topic, subject that the the entity under study has been properly identified, and the encoding of this entity has no errors.

Q: How do we evaluate competing scientific theories?

There exists multiple methods for the comparison of competing scientific theories. Karl Popper's falsificationism is a well known one. According to Popper, scientific theories must be falsifiable, that is, they must be so clearly stated and precise that we can validate them by means of performing an experiment. The more precisely a theory is formulated, and the more accurate its predictions, the better, since that increases the chances of being falsified. As long as the experiments confirm the theory we keep it as valid. However, if a single experiment fails, the theory should be rejected. The main problem of falsificationism is that, in practice, if a experiment fails to confirm a theory, we can not be sure about what went wrong, the experiment or the theory. In the theory

of nescience we completely agree with Popper's idea that theories must be clearly stated. In fact, our theories, being Turing machines, are as clearly stated as possible. But, on the contrary of what falsificationism proposes, we do not reject a theory because it has been falsified with an experiment, instead what we propose is to measure the error produced, and take that error into account in our measure of how good is the theory, that is, the nescience of our current best description. Another difference is that, in principle, a theory that makes better prediction is not automatically preferred to another one with less predictability, since we have to take into account not only the error, but also the redundancy of the theory. Unfortunately, in practice, the theory of nescience suffers the same problem than falsificationism, since given that the error of a description must be estimated in practice, usually by means of performing an experiment, we can not be sure if the error is due to the incorrectness of the theory, or due to a wrong designed experiment.

Other alternative interpretations of science, with a stronger orientation towards the theoretical frameworks in which scientific activities take place, like Kuhn's scientific paradigms or Lakatos' research programs, do not have a clear interpretation in the context of the theory of nescience. The same happens when we take into account the social or political aspects of science. Please mind that we are not saying that these possible explanations of science are incorrect. In fact, we have taken seriously the recommendations of Popper, Kuhn, Lakatos and many other philosophers of science during the development of our own theory of nescience.

Q: Is there a universal scientific method?

Although we provide a methodology for the discover of interesting questions, a method to discover what it is hidden in the unknown area, the theory of nescience does not, and does not intend, to be a method (or methodology) for the development of new science. However, in the framework provided by the theory of nescience we could address the problem of which one of the scientific methods proposed so far is more effective to discover new theories. By means of an historical analysis, and by means of analyzing how well the evolution of error and redundancy is covered by the method.

TODO: How non-falsifiable theories are managed by the theory of nescience? What about the desirable property of being as much falsifiable as possible?

Nescience provides a tool that can be applied to all disciplines.

Feyerabend argues that there is no such method, he proposes the principle that "anything goes". anarchistic account of science [...] "increases the freedom of scientists by removing them from methodological constraints"

Q: Does science enable progress?

TODO: Generality of a theory (the amount of things it can explain), and the problem of too generic theories. Is there somethin like "information content" of a theory

Popper suggested to propose highly speculative, highly falsifiable, theories. That it is in line with our methodology to the discovery of new research topics. According to Popper we cannot say that a theory is true, just that it is the best, yet non falsified, theory. New theories must be more falsifiable than the ones they replace. An ad-hoc modification that does not decrease the error, will increase the nescience because it increases the redundancy. However, to Popper, a prediction is novel if it involves a phenomenon that does not figure in the background knowledge of the time.

"According to Kuhn, science progress following the following schema: pre-science, normal science, crisis, revolution, new normal science, new crisis [...] There will be no purely logical argument that demonstrates the superiority of one paradigm over another and that thereby compels a rational scientist to make the change [...] proponents of rival paradigms will subscribe to different sets of standards and metaphysical principles [...] rival paradigms are incommensurable."

Q: What is the difference between science and pseudoscience?

As it was pointed out by Feyerabend, all the different proposals made by philosophers to identify that remarkable method that makes science possible (logical positivist, falsificationism, theory-dependent models, etc.) failed even to explain a classical example like the Copernican revolution. According to Feyerabend, there is no such thing as a scientific method; instead, he proposed that in science, anything goes. Moreover, Feyerabend asserts that there is nothing special about science that makes it superior to other forms of knowledge. We completely disagree with this anarchist point of view. As we will see in detail in Section 19.7, the theory of nescience provides a clear answer to the question of what distinguishes science from pseudoscience. According to our theory, in science we have that nescience decreases when we invest an amount of effort to gather new knowledge, however, in pseudosciences, nescience does not decrease in spite of the effort invested.

Unfortunately, this capacity of decrease nescience given effort is a necessary condition to classify a knowledge discipline as scientific, but not a sufficient one. That is, not all disciplines in which nescience decreases with effort are classified as science according to our intuitive idea of what it is science. For example, **XXX** is a case of a discipline in which nescience decreases with time but it is not considered to be a science.

19.7 Evolution of Knowledge

TODO: Show the evolution of a research topic, preferably by using historical data from centuries ago, if not possible, using the evolution of a topic given wikipedia.

■ **Example 19.1** In Figure 19.6 is depicted the evolution of a hypothetical research topic. Every point in the graph represents a new current best description for that topic (descriptions are ordered with respect to time). New descriptions could be based on novel theories that explain the topic, refinements over already existing ones, or on a reduction of the number of assumptions. In general, each new description should be shorter than the previous one. However, as we can see in the figure, it might happen that some descriptions are longer than its previous. That could be the case, for example, when we discover new facts that have to be taken into account by the theory. But the important thing is that nescience should decrease, in average, with time. ■

My hypothesis is that the nescience of topic descriptions decrease with time. On the contrary, the description of topics from pseudoscience or religion does not decrease with time. This property, decrease with time is what distinguish science with respect other pseudosciences. I totally disagree with Feyerabend when he states that *sience has no special features that render it intrinsically superior to other kinds of knowledge such as ancient myths or voodoo*.

How does my view differ with respect to the point of view that states that "science is derived from facts"? Perhaps facts is what allow us to make shorter descriptions. But facts also can increase the length of a description.

Here I have to be very careful, because if the description lenght of topics in phylosophy does not decrease with time, we should conclude that phylosophy is a pseudoscience.

In order to understand a theory a background knowledge is assumed. What it is exactly that background is something that depends on the theory and the current understanding of the theory.

■ **Example 19.2** Suppose that the topic t being studied is the concept of *limit of a function*. The standard (ε, δ) -definition of limit provided by Karl Weierstrass is:

$$\lim_{x \rightarrow c} f(x) = L \Leftrightarrow \forall \varepsilon > 0 \exists \delta > 0 : \forall x (0 < |x - c| < \delta \Rightarrow |f(x) - L| < \varepsilon)$$

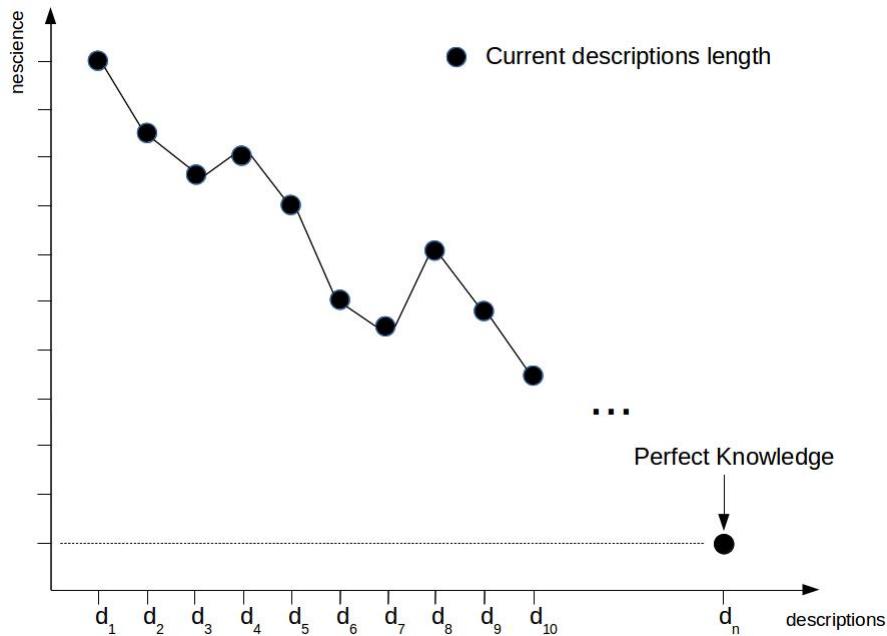


Figure 19.6: Evolution of Nescience

This definition has a length of 46 symbols, spaces not included². The alternative *infinitesimal definition* of limit, based on non-standard calculus, is:

$$\lim_{x \rightarrow c} f(x) = L \Leftrightarrow st(x) = c \Rightarrow st(f(x)) = L$$

where $st(\cdot)$ denotes the *standard part function*, which "rounds off" each finite hyperreal³ number to the nearest real. This second definition has a length of 31 symbols, and so, we say that our nescience of the concept of limit has decreased, since we were able to simplify the definition from a three quantifier block to a just one quantifier statement.

If the complexity C_t of the concept of limit is less than 31 characters, then there must be possible to come up with an even shorter definition. □ ■

19.8 Graspsness of Topics

In Section ref? we saw a measure of how difficult it is to grasp a research area. In this section we are going to see a measure of how difficult it is to grasp an individual research topic.

TODO: Study the graspsness of a research topic difficult to understand, and the graspsness of a easy to understand topic

19.9 Probability of Being True

TODO: Study the probability of being true.

²

■ **Example 19.3** For the sake of simplicity, we have computed the complexity of the topic given the number of symbols, not the the length of its binary representation, as it is required by our definition. ■

³Nonstandard analysis deals primarily with the pair $\mathbb{R} \subset^* \mathbb{R}$, where hyperreals ${}^*\mathbb{R}$ are an ordered field extension of the reals \mathbb{R} , that contains infinitesimals in addition to reals.

TODO: If the acceptability of experimental results is theory-dependent, how can we change the probability of being true of a theory in the presence of new experimental results? There is a high risk of circular argument in the bayesian interpretation of science.

The Bayesian interpretation of science, based on the Bayes theorem of conditional probabilities, states that the probability that an hypothesis H is true given that an experiment E has been successful, $P(H|E)$, is given by:

$$P(H|E) = P(H) \frac{P(E|H)}{P(E)}$$

where $P(H)$ is the prior probability of H , $P(E|H)$ is the probability of being the experiment successful given that the hypothesis is true, and $P(E)$ is the probability of being true the experiment. In this way, Bayes is a continuous process in which every successful experiment will increase the probability of H being true, and a failed experiment ([How?](#)) will decrease the probability. The very initial probability $P(H)$, prior to any experiment, is given by the length of its description:

$$P(H) = r^{l(H)}$$

■ **Example 19.4** **TODO:** A real example, with data ■

TODO: How my theory of nescience integrates with this?

TODO: How the base knowledge a affect the probability of the hypothesis H ?

TODO: How can we use Bayes and what-if scenarios to improve an hypothesis?

19.10 Unused text

Here I should mention the work of Solomonoff with respect to assign a prior probability to theories. I what Solomonoff proposed was to use the shortest length of programs, what I propose is essentially different, since what I use is lengths of descriptions.

The same that we do not try to define what it is a research topic, I do not try to provide an explanation of from where scientific theories come from. According to my interpretation, anything could be a theory, with a initial probability of being true. However, we, as scientists, could clean-up those very unlikely of being true theories.

I will show that the distinctive element of scientific knowledge is that with time nescience decreases, as opposed to other kinds of knowledge, such as ancient myths or voodoo, where nescience does not decrease with time.

Scientific knowledge can neither be conclusively proved nor conclusively disproved, but we can assign a probability of being true, a probability that will change with time, as new facts are discovered, and new experiments are performed.

Theories are based on the observation of facts and experimentation. Both, the observation of facts and the experiment design are based on our current knowledge. If we have a defective knowledge, our theories would be defective. New knowledge, or new technologies, allows us to gather new facts, design new experiments, and define new theories.

Current theories will be eventually replaced in the future by new ones.

Theories are based on other theories and relate to other theories, forming a complex network of interrelations between theories.

References

The behavior of compressors depending of the size of objects and window (buffer) size is studied in detail in [[cebrían2005common](#)] with applications to the normalized compression distance (a measure of similarity between objects proposed in [[li2004similarity](#)]).



20. Computational Creativity

*To be surprised, to wonder,
is to begin to understand.*

José Ortega y Gasset

In this Chapter we are going to see how to apply in practice our methodology for the assisted discovery of interesting research questions. As it was the case of previous chapter, in which we studied the concept of nescience from a practical point of view, ...

In the first part of this chapter we will see how to approximate the new metrics introduced: relevance and applicability. The relevance of a topic will be based on the number of web pages on Internet that link to the topic's page on Wikipedia (external links), and applicability will be estimated by the number of links from the Wikipedia's scientific pages to themselves (internal links). We will provide some practical examples of both quantities for the set of topics that compose the research area of theoretical computer science. Then, we will describe how to apply in practice our methodology for the discovery of interesting questions, and we will come up with some examples of new research questions that, in principle, could be addressed by science. The new questions proposed will be both, intradisciplinary, coming from the area of theoretical computer science, and interdisciplinary, by means of combining the area of theoretical computer science with the area of philosophy and the area of biochemistry. Finally, we will derive some new interesting research topics, according to our subjective interpretation of the combinations found, that are enough interesting to deserve to be the subject of new research activities. We will also evaluate if the proposed topics fulfill the requirements that we proposed in Chapter 15 for a question to be classified as interesting.

In the last part of the chapter, we will apply the set of metrics defined for the classification of individual research topics to full research areas. In this way, we will compute the interestingness of the different research disciplines as source of new problems, and their interestingness as a source of useful tools to solve open problems. These metrics will allow us to compare the relative merits of different knowledge disciplines. Some examples of research areas in decay will be shown as well.

20.1 Classification of Research Topics

In order to evaluate the classification metrics proposed we have used the set of topics corresponding to all the categories at Level 4, from the category Level 3 *theory of computation* (included in category Level 2 *theoretical computer science* and category Level 1 *formal sciences*).

Before to compute the new interesting questions, it is highly convenient to normalize the metrics of the topics involved in the study, otherwise, a very reduced set of topics could dominate all the questions. For the normalization process we have used the BoxCox method, that it is based on the identification of the best transformation from a family of power transformations.

TODO: Explain the BoxCox method

Also, and since these studied quantities, relevance, applicability, nescience and maturity, do not have the same scale, it is highly convenient to apply the following additional transformation:

$$\mu_t = \frac{\mu_t - \min(\mu)}{\max(\mu) - \min(\mu)}$$

where μ_t refers to the considered metric (nescience, relevance, ...).

20.1.1 Relevance

In Definition 15.1.2 we introduced the concept of relevance of a research topic as a measure of the impact that this topic could have in people's life. The idea was that the higher the relevance, the higher its potential as a source of interesting questions, since we would be addressing a problem that affects many people. Relevance was defined as the degree of the research topic in the relevance graph, a bipartite graph connecting topics and people (see Definition 15.1.1). Of course, this relevance graph is a mathematical abstraction that it is very difficult to compute in practice, since we do not have information about how people is affected by each topic.

As an approximation of the relevance of a topic we have used the number of links (URLs) from external web pages on the whole Internet that point to the topic's web page on Wikipedia. The rationale is that the more relevant is a topic, the more people will be talking about it on Internet, and the more URLs there will be linking to Wikipedia, since Wikipedia is a well known source of information to which many people refer. In fact, we are not interested in knowing the absolute relevance of research topics, since what we need is a measure of relative relevance between different topics. An underestimate of the relevance is not harmful as long this underestimation is equally distributed among all the topics. It requires further research to fully understand how well the theoretical concept of relevance is approximated by this URLs counting procedure.

Another problem is that we do not know the number of links to a web page on Internet, and so, we have to apply a second approximation. In this case, the number of links was estimated using Google's link: facility in searches. Google's link: lists the links that Googlebot discovered during its crawling and indexing process of Internet. For example, the number of external pages that link to the *Computer Science* article in Wikipedia is given by:

```
link:http://en.wikipedia.org/wiki/Computer_science
```

As Google recognizes in its web page, not all links to a site may be listed. The number of links could vary due to redirections, crawling errors, and other problems. How these errors affect to the accuracy of the links counting is not clear, since the details of the Google's crawling algorithm are not public.

Figure 20.1 shows a plot of the relevance of the set of topics after the application of the normalization process described in XXX. A practical problem is that there are too many topics with a very low number of links (as indexed by Google). That should not be a problem since we are not using at any moment those topics with very low relevance.

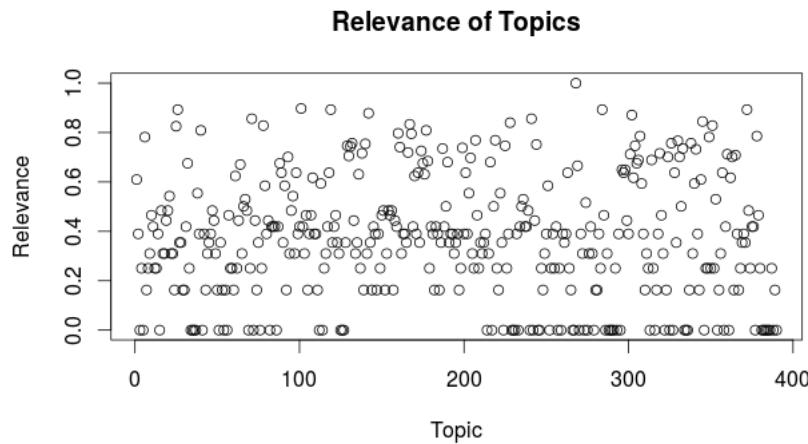


Figure 20.1: Relevance of topics

Topic	Relevance	Norm.
Regular expression	409	1.00
Turing machine	159	0.90
Binary number	141	0.89
Recursion	133	0.88
Finite-state machine	118	0.86
Halting problem	108	0.85
Cellular automaton	104	0.85
Floating point	99	0.84
Lambda calculus	95	0.84
Turing completeness	93	0.83

Table 20.1: Relevance of topics

Table 20.1 contains the ten most relevant topics according to its relevance. For each topic it is shown its relevance (number of external links) and the normalized version of this number. The list includes basic concepts (*binary number*, *floating point*), advanced concepts (*Turing machine*, *finite-state machine*, *cellular automaton*, *lambda calculus*, *Turing completeness*), highly popular tools (*recursion*, *regular expression*) and classical problems (*halting problem*). All those topics could fit into our intuitive idea of highly relevant, although some authors could perfectly disagree that some of them are the most relevant ones in the area of theory of computation (for example, *floating point*).

20.1.2 Applicability

Applicability measures how likely is that a research topic can be applied to solve open problems. If a tool has been already applied to solve multiple problems, then there is a high probability that it can be used again to solve new problems. The number of problems in which a tool has been applied is computed with the aid of the applicability graph (see Definition 15.2.1), and applicability is formally defined as the out-degree of the topic in this graph (see Definition 15.2.2). We have approximated the applicability graph by means of using the graph of internal links between the scientific pages of Wikipedia. That is, we approximate the applicability of a topic by counting the number of pages from Wikipedia domain that links to the topic's page (we have used the “What

links here" facility from Wikipedia, a tool to see the list of the pages that link to, or redirect to, the current page). As it was the case of the relevance of a topic, we are not interested in the absolute value of the applicability of topics. What it is important for us is the relative ordering of topics based on their applicability. Perhaps, a better approach to approximate the applicability of a topic would be to analyze the graph of citations from academic research papers, combined with the automatic identification of the topics addressed in those papers.

The applicability of a topic Figure 20.2 shows a plot of the applicability of the selected set of topics after the normalization process, and Table 20.2 contains the ten most relevant topics according to its applicability. For each topic it is shown the number of internal links and the normalized version of this number. In the list we can find topics like *regular expression*, *recursion*, *Turing machine*, or *cellular automaton* that perfectly fits our intuitive idea of tools that can be applied to solve other problems. However, we can also find in the list the topic *quantum computer* that is not very applicable, but since it is a highly popular research topic it is mentioned in many different Wikipedia's pages. Other pages like *division by zero*, *floating point*, or *binary number*, do not seem to be good tools either. The final topic, *computability theory*, shows that, even at the fourth level, we still can find too broad topics.

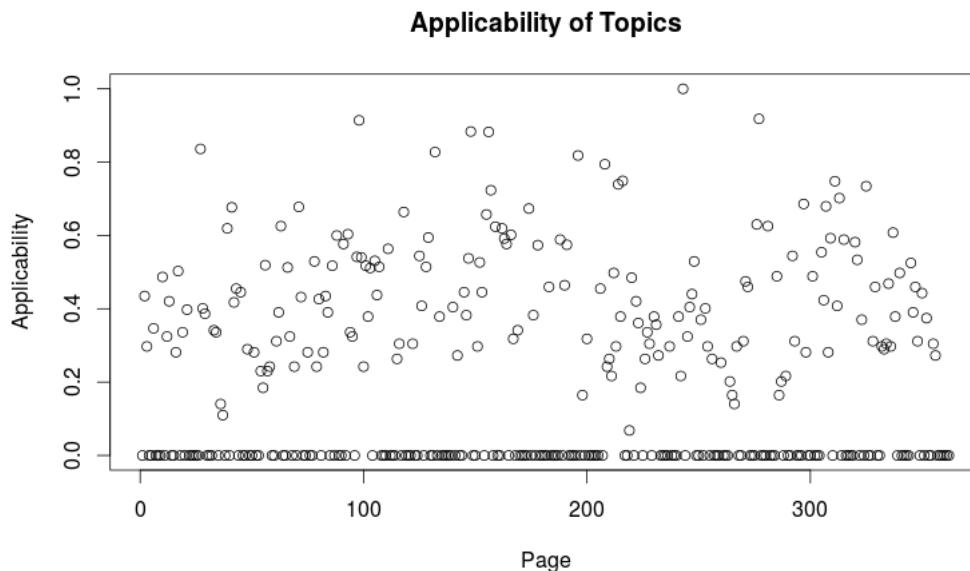


Figure 20.2: Applicability of topics

20.1.3 Maturity

20.2 Interesting Research Questions

20.2.1 Intradisciplinary Questions

20.2.2 Interdisciplinary Questions

20.3 New Research Topics

20.4 Classification of Research Areas

20.5 References

Papers about the BoxCox method ...

Topic	Applicab.	Norm.
Regular expression	1971	1.00
Recursion	1227	0.91
Quantum computer	1197	0.91
Division by zero	998	0.88
Floating point	992	0.88
Turing machine	748	0.83
Binary number	710	0.82
Ternary num. sys.	670	0.80
Cellular automaton	433	0.74
Computability theory	430	0.73

Table 20.2: Applicability of topics

20.6 Future Work

Appendix



A	Linear Algebra	309
A.1	Vectors	
A.2	Matrices	
B	Foundations of Mathematics	311
B.1	Propositional Logic	
B.2	Predicate Logic	
B.3	Set Theory	
B.4	Lambda Calculus	
B.5	Category Theory	
C	Advanced Mathematics	319
C.1	Distinctiveness of Metrics	
C.2	Distinctiveness of Metrics	
D	Coq Proof Assistant	325
E	About Quotes and Photos	329
F	Notation	335
	Bibliography	339
	Books	
	Articles	

A. Linear Algebra

All great work is the fruit of patience and perseverance,
combined with tenacious concentration on a subject
over a period of months or years.
Santiago Ramón y Cajal

TODO: Pending

A.1 Vectors

A.2 Matrices

A *matrix* A of order $m \times n$ is a set of mn scalars ordered in m *rows* and n *columns* in the following way:

$$A = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix}$$

The *entry* a_{ij} corresponds to the element of A located at row i and column j . We denote by $\mathcal{M}_{m \times n}$ the *set of all matrices* of order $m \times n$. A *row matrix* is a matrix that belongs to $\mathcal{M}_{1 \times n}$, and a *column matrix* to $\mathcal{M}_{m \times 1}$. A *squared matrix* is a matrix that belongs to $\mathcal{M}_{n \times n}$. The entries a_{ii} form the *main diagonal* of a square matrix. The *transpose matrix* of a matrix $A \in \mathcal{M}_{m \times n}$ is the matrix $A^T \in \mathcal{M}_{n \times m}$ whose entries (i, j) are equal to the entries (j, i) of A . If $A = A^T$ we say that A is a *symmetric matrix*. A *diagonal matrix* is a matrix whose all its entries outside the main diagonal are zero. The *identity matrix* I is a diagonal squared matrix with all the entries in the main diagonal equal to 1. A *submatrix* of a matrix is obtained by deleting any collection of rows and/or columns.

■ **Example A.1** Given the squared matrix $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$ we have that the entry $(2, 3)$ has the value 6, the diagonal is composed by the numbers 1, 5, and 9, its transpose is $A^T = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$, and that the matrix $B = \begin{pmatrix} 1 & 3 \\ 4 & 6 \end{pmatrix}$ is a submatrix of A . ■

The sum of two matrices A and B of equal size is the matrix $A + B$ whose entry (i, j) are $(A + B)_{ij} = a_{ij} + b_{ij}$. The operation of sum of matrices is associative $(A + B) + C = A + (B + C)$, commutative $A + B = B + A$, has a neutral element $A + 0_{m \times n} = A$ and an inverse element $A + (-A) = 0_{m \times n}$. The multiplication of a scalar λ by a matrix A is the matrix λA whose entry (i, j) is $(\lambda A)_{ij} = \lambda a_{ij}$. The operation of multiplication of a scalar by a matrix is distributive with respect to the sum of matrices $\alpha(A + B) = \alpha A + \alpha B$, distributive with respect to the product by scalars $(\alpha + \beta)A = \alpha A + \beta A$, associative with respect to the product by scalars $(\alpha\beta)A = \alpha(\beta A)$, and has a unit element $1A = A$. The product of two matrices $A_{m \times n}$ and $B_{n \times p}$ is the matrix $AB_{m \times p}$ whose entry (i, j) is $(AB)_{ij} = \sum_{k=1}^n a_{ik}b_{kj}$. The operation of multiplication of matrices is associative $(AB)D = A(BD)$, has a left neutral element $AI_n = A$ and a right neutral element $I_m A = A$, associative with respect to the product by scalars $\alpha(AB) = (\alpha A)B = A(\alpha B)$, distributive with respect the sum of matrices by the right $A(B + C) = AB + AC$ and the left $(B + C)D = BD + CD$. Additionally, the operation of transpose satisfies the following properties: $(A + B)^T = A^T + B^T$, $(\lambda A)^T = \lambda A^T$, $(AB)^T = B^T A^T$.

■ **Example A.2** Given the matrices $A = \begin{pmatrix} 1 & 2 \\ 3 & 5 \end{pmatrix}$ and $B = \begin{pmatrix} 5 & 6 \\ 7 & 8 \end{pmatrix}$ we have that $A + B = \begin{pmatrix} 6 & 8 \\ 10 & 13 \end{pmatrix}$, $2A = \begin{pmatrix} 2 & 4 \\ 6 & 10 \end{pmatrix}$ and $AB = \begin{pmatrix} 19 & 22 \\ 50 & 58 \end{pmatrix}$. ■

A square matrix A is *invertible* or *non-singular* if there exists a matrix B such that $AB = BA = I$. If A is non-singular, B is unique and is called the *inverse matrix* of A , denoted by A^{-1} . A matrix A is *orthogonal* if its transpose is equal to its inverse $A^T = A^{-1}$; the columns and rows of a orthogonal matrix are called *orthonormal vectors*.

The determinant is a scalar value that is a function of the entries of a square matrix denoted $\det(A)$

Leibniz formula.

The determinant is nonzero if and only if the matrix is invertible

rank

A number λ , and a non-zero vector \mathbf{v} satisfying $A\mathbf{v} = \lambda\mathbf{v}$ are called an *eigenvalue* and an *eigenvector* of A , respectively.

■ **Example A.3** The determinant of a 2×2 matrix is and the determinant of a 3×3 matrix is Refer to the References section ■

Matrix decomposition is the process to render a matrix into a more easily accessible form, meanwhile certain properties, like the determinat or the rank, are preserved. The *singular value decomposition* of a matrix A of order $m \times n$ is a factorization of the form $A = U\Sigma V^T$ where U is an $m \times m$ orthogonal matrix, Σ is an $m \times n$ non-negative rectangular diagonal matrix, and V^T is the traspose of an $n \times n$ orthogogal matrix.

References

TODO: Pending.

B. Foundations of Mathematics

TODO: Introduce this chapter.

Propositional logic deal with sentences, which can be true or false, and they way we can combine those sentences to construct arguments. Predicate logic extends propositional logic introducing quantified variables and relations, allowing richer arguments. Logic is studied here with the aim of providing the tools to examine the soundness, consistency and completeness of the axiomatic foundations of the theory of nescience. The exposition of logic contained in this chapter is a little bit unconventional. Most of the textbooks describing logic assume set theory, and the books on set theory require logic. We avoid this circularity by defining formulas as strings of symbols and logic as the rules to properly manipulate those strings. With this approach we loose the intuition behind logic, and the interpretability of the results. However, this book is about computers and automation, where meaning does not play a particular role.

TODO: Explain why if we have predicate logic to formalize the theory of nescience we also include type theory and category theory in this chapter.

B.1 Propositional Logic

Propositional logic is about the relationship between the truth of *propositions*. Most of the authors require propositions to be declarative sentences, that is, they can be affirmed or denied. However, from a formal point of view, the content of propositions is not relevant to logic, and so, we will assume that any finite string of symbols can be a proposition.

The language (syntax) of propositional logic contains two *primitive symbols*, \neg and \wedge , called "not" and "and" respectively. Propositions that do not contain those primitive symbols are called *atomic sentences*. Atomic sentences are denoted by capital letters A, B, C, \dots .

Formulas in propositional logic are derived from atomic sentences and primitive symbols.

Definition B.1.1 A finite string of symbols is a *formula*, denoted by the capital letters F, G, H, \dots , if and only if it is an atomic sentence or it is build up from atomic sentences by repeated application of the following two rules:

R1 (negation) If F is a formula, then $\neg F$ is a formula.

R2 (conjunction) If F and G are formulas, then $(F \wedge G)$ is a formula.

For example, the string $\neg(A \wedge (B \wedge C))$ is a formula, but the string $\neg \wedge (A \wedge BC)$ is not.

In order to make complex formulas easier to read (by humans), the following convenient abbreviations are introduced:

Definition B.1.2 Let F and G be two formulas. The symbols \vee , \rightarrow and \leftrightarrow , called "or", "implies" and "if and only if" respectively, are defined as:

- (i) $(F \vee G)$ abbreviates $\neg(\neg F \wedge \neg G)$
- (ii) $(F \rightarrow G)$ abbreviates $(G \vee \neg F)$
- (iii) $(F \leftrightarrow G)$ abbreviates $(F \wedge \rightarrow G) \wedge (G \rightarrow F)$

Also, with readability in mind, the following notation is used:

Notation B.1. If F or (F) is a formula, then the formulas F and (F) are considered as the same formula.

If no parentheses are present, the symbol \neg has priority over the symbols \wedge , \vee , \rightarrow and \leftrightarrow . For example, the formulas $\neg(A \wedge B)$ and $\neg A \wedge B$ are different.

A subformula of a formula F is a substring of F that is itself a formula.

Definition B.1.3 The *subformulas* of a formula are defined as:

- (i) If F is a formula, then F is a subformula of itself.
- (ii) If F is a formula and G is a subformula of F , then G is also a subformula of $\neg F$.
- (iii) If F and G are formulas and H is a subformula of F or a subformula of G , then H is also a subformula of $(F \wedge G)$.

From a semantic point of view, each formula has either a true value of *true* or a true value of *false*, represented by the symbols 1 and 0 respectively. The semantic of a formula is derived using *true tables*. A true table is a table that given an assignment of values to atomic formulas allows us to derive the true value of more complex formulas.

The true table of the not symbol \neg is given by:

A	$\neg A$
0	1
1	0

The true table of the and symbol \wedge is given by:

A	B	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

The true tables of more complex formulas can be derived by the repeated application of these two tables.

■ **Example B.1** The true table of the if-then, or implies, symbol \rightarrow can be derived as follow:

A	B	$\neg A$	$\neg A \vee B$
0	0	1	1
0	1	1	1
1	0	0	0
1	1	0	1

As we can observe in this table, a counterintuitive property of propositional logic is that a false statement implies anything. ■

Similar tables can be derived for the symbols \vee and \leftrightarrow .

Let A_1, \dots, A_n be a list of n atomic sentences, and let $\mathcal{F}(A_1, \dots, A_n)$ denote all the formulas that can be built from those atomic sentences given Definition B.1.1. The list A_1, \dots, A_n is denoted by S .

Definition B.1.4 An *assignment* of S , denoted by $\mathcal{A}(S)$ or simply \mathcal{A} if there is no doubt about the S to which we are referring, is a mapping of the sentences of S to a list of n true values.

An assignment $\mathcal{A}(S)$ can be extended to all the sentences of $\mathcal{F}(S)$ by means of computing the corresponding true tables. The true value of a sentence F given the assignment $\mathcal{A}(S)$ is denoted by $\mathcal{A}(F)$.

Definition B.1.5 If $\mathcal{A}(F) = 1$ then it is said that F holds under assignment \mathcal{A} , or that \mathcal{A} models F , and it is denoted by $\mathcal{A} \models F$. A formula is *satisfiable* if it holds under some assignment. A formula is *valid*, denoted by $\models F$, if it holds under every assignment; a valid formula is called a *tautology*. A formula is *unsatisfiable* if it holds under no assignment; an unsatisfiable formula is called a *contradiction*.

■ **Example B.2** Next table shows that the formula $(A \vee B) \wedge (\neg A \wedge \neg B)$ is a contradiction.

A	B	$A \vee B$	$\neg A$	$\neg B$	$\neg A \wedge \neg B$	$(A \vee B) \wedge (\neg A \wedge \neg B)$
0	0	0	1	1	1	0
0	1	1	1	0	0	0
1	0	1	0	1	0	0
1	1	1	0	0	0	0

Definition B.1.6 Let F and G be two formulas. It is said that G is a *consequence* of F , denoted by $F \models G$, if, and only if, $F \rightarrow G$ is a tautology,

It is easy to show that a formula G is a consequence of formula F if for every assignment \mathcal{A} , if $\mathcal{A} \models F$ then $\mathcal{A} \models G$.

Definition B.1.7 Let F and G be two formulas. It is said that F and G are *equivalent*, denoted by $F \equiv G$, if, and only if, $F \leftrightarrow G$ is a tautology.

It is easy to show that if G is a consequence of F and F is a consequence of G , then F and G are equivalent.

■ **Example B.3** The *law of contraposition* states that A implies B is equivalent to not B implies not A ; with symbols $(A \rightarrow B) \equiv (\neg B \rightarrow \neg A)$. Next true table shows that $(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$ is a tautology.

A	B	$A \rightarrow B$	$\neg B$	$\neg A$	$\neg B \rightarrow \neg A$	$(A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)$
0	0	1	1	1	1	1
0	1	1	0	1	1	1
1	0	0	1	0	0	1
1	1	1	0	0	1	1

Introduce and explain the following definition.

Definition B.1.8 Let $\mathcal{F} = \{F_1, F_2, \dots\}$ be a set of formulas. For any assignment \mathcal{A} , we say \mathcal{A} models \mathcal{F} , denoted by $\mathcal{A} \models \mathcal{F}$ if $\mathcal{A} \models F_i$ for each formula $F_i \in \mathcal{F}$. We say a formula G is a consequence of \mathcal{F} , and write $\mathcal{F} \models G$, if $\mathcal{A} \models \mathcal{F}$ implies $\mathcal{A} \models G$ for every assignment \mathcal{A} .

A proof system consists of a set of basic rules for derivations. These rules allow us to deduce formulas from sets of formulas. It may take several steps to derive a given formula G from a set of formulas \mathcal{F} , where each step is an application of one of the basic rules. The list of steps forms a formal proof of G from \mathcal{F} .

We write $\mathcal{F} \vdash G$ to abbreviate " G can be derived from \mathcal{F} ".

A proof system is sound if the following property hold: if a formula G can be derived from a set of formulas \mathcal{F} , then G is a consequence of \mathcal{F} (if $\mathcal{F} \vdash G$, then $\mathcal{F} \models G$).

If a proof system is sound, then it provides an alternative to truth tables for determining whether a formula G is a consequence of a set of formulas \mathcal{F} .

Definition B.1.9 A formal proof in propositional logic is a finite sequence of statements of the form " $\mathcal{X} \vdash Y$ " (where \mathcal{X} is a set of formulas and Y is a formula) each of which follows from the previous statements by one of the rules in Table 1.5. We say that G can be derived from \mathcal{F} if there is a formal proof concluding with the statement $\mathcal{F} \vdash G$.

Formulas F and G are provably equivalent if both $\{F\} \vdash G$ and $\{G\} \vdash F$. If F and G are provably equivalent, then they are equivalent.

TODO: Proofs in the logic and proofs outside the logic.

B.2 Predicate Logic

Predicate logic, also known as *first-order logic*, extends propositional logic with the use of variables and quantifiers over variables, in such a way that we can state the relation between non-logical objects. The language of predicate logic contains three *primitive symbols*, the two symbols inherited from propositional logic \neg , \wedge , and a new symbol \exists called *exists*. In this book we are interested in predicate logic with equality, and so, we add a fourth primitive symbol $=$ called *equality*.

In propositional logic our object of study were formulas, that is, strings of symbols that follows well-defined syntactic rules. In predicate logic we also deal with formulas. The difference is that some of the symbols contained in formulas are distinguished because they play a special role. The distinguished symbols are :

Variables denoted by lower case letters from the end of the alphabet (\dots, x, y, z).

Constants denoted by lower case letters from the beginning of the alphabet (a, b, c, \dots).

N-ary functions denoted by lower case letters f, g and h .

N-ary relations denoted by upper case letters P, Q, R and S .

Notation B.2. Given the limited number of letters of the alphabet, we can use subscripts to distinguish between the elements of formulas. For example, x_1, x_2, x_3, \dots

The concept of N-ary function and N-ary relation is better understood in the context of set theory. However, at this point we can not talk about sets, since the concept of set has not been formally defined yet. As we will see in Section B.3, in order to deal with sets, we have first to introduce a new symbol \in , and assume that some axioms (which are based on predicate logic) are true. By the moment, functions and relations are just distinguished symbols.

Next definition introduces the concept of *term*, an intermediate step before to define the concept of *formula*.

Definition B.2.1 A finite string of symbols is a *term* if and only if it is built up by repeated application of the following two rules:

T1 Every variable and every formula is a term.

T2 If f is an N -ary function and t_1, t_2, \dots, t_N are terms, then $f(t_1, t_2, \dots, t_N)$ is also a term.

Given the concept of term, we can formally define what we mean by atomic formula.

Definition B.2.2 Let t_1 and t_2 be two terms, then the string $t_1 = t_2$ is an *atomic formula*. Let R be a relation and t_1, t_2, \dots, t_N be N terms, then the string $R(t_1, t_2, \dots, t_N)$ is an *atomic formula*.

Formulas are built based on primitive symbols and atomic formulas. Recall rules **R1** and **R2** of Definition B.1.1. The same rules can be applied in predicate logic, together with a new rule. We use lower case Greek letters to denote formulas.

Definition B.2.3 A string of symbols is a formula of first order logic if it is an atomic formula, or it can be derived by repeated application of the following rules:

R1 If φ is a formula, then $\neg\varphi$ is a formula.

R2 If φ and ψ are formulas then $\varphi \wedge \psi$ is a formula.

R3 If φ is a formula and x a variable then $\exists x\varphi$ is a formula.

In predicate logic we also introduce the following convenient abbreviation.

Definition B.2.4 Let x be a variable and φ a formula. The symbol \forall , called *for all*, is defined as:

1 $\forall x\varphi$ abbreviates $\neg\exists x\neg\varphi$.

■ Example B.4 ■

If no parenthesis are present, \neg , \exists and \forall have priority over \wedge , \vee , \rightarrow and \leftrightarrow .

In order to define the semantics of the symbols \exists and $=$ we need the concept of *structure*, since those symbols only make sense when they are associated with a structure. A structure consists of a underlying set together with an interpretation of the constants, functions and relations. However, in order to that, first we have to formally define what we mean by set.

B.3 Set Theory

In this section we describe the ZFC axiomatic system for the theory of sets, named after the mathematicians Ernst Zermelo and Abraham Fraenkel. The 'C' in ZFC refers to the axiom of choice, also included in the initial list of axioms. ZFC is considered the standard form for axiomatic set theory, and it is the most common foundation of mathematics. An important point of ZFC is that all the entities in the universe of discourse are sets, and so, elements that are not sets themselves are not allowed.

Axiom of Extensionality If the set x and the set y have the same elements, then $x = y$.

Axiom of Pairing For any sets a and b there exist a set $\{a, b\}$ that contains exactly a and b .

Axiom Schema of Separation If P is a property with parameter p , then for any set x and parameter p there exists a set $y = \{u \in x : P(u, p)\}$ that contains all those sets $u \in x$ that have property P^1 .

Axiom of Union For any set x there exists a set $y = \cup x$, the union of all elements of x .

Axiom of Power Set For any set x there exists a set $y = \mathcal{P}(x)$, the set of all subsets of x .

Axiom of Infinity There exists an infinite set.

Axiom Schema of Replacement If F is a function, then for any set x there exists a set $y = F(x) = \{F(u) : u \in x\}$.

Axiom of Regularity Every nonempty set has an \in -minimal element.

¹It called axiom schema because there exists one axiom for each P .

Axiom of Choice Every family of nonempty sets has a choice function.

B.4 Lambda Calculus

Lambda calculus is a formal system for the foundations of mathematics. Lambda calculus is based on the concepts of abstraction, application and reduction of functions, and it provides a characterization of what it is a computable function. In this sense, lambda calculus constitutes a universal model of computation, and so, it is equivalent to a (particular) universal Turing machine.

We start by assuming the existence of a finite list of symbols v_1, v_2, \dots, v_n called *variables*, a symbol λ called *abstractor*, and the parenthesis $(,)$. The list v_1, v_2, \dots, v_n is not a set, in the sense of Section B.3, since the symbol \in is not defined, and the ZFC axioms are not required to be true. The concept of *list* has to be understood with its usual meaning in natural language, in the same way that the words *assume*, *exists* and *finite* are also understood. Type theory is not based on predicate logic either, since theorems can be proved within the theory itself, without requiring external elements. However, we will use predicate logic in our description of the theory, because it allows us to be very clear and concise in definitions and propositions.

Definition B.4.1 A λ -term is defined inductively by:

$$\begin{aligned} V &:= v_1 \mid v_2 \mid \dots \mid v_n \\ T &:= V \mid (T T) \mid (\lambda V. T) \end{aligned}$$

Definition B.4.1 encapsulates the way that functions are constructed. We have an *application* procedure that given the λ -terms M and N builds a new term $M N$ in which M is applied to N , and there is an *abstraction* procedure in which a variable v is abstracted from the term M .

We denote Λ the collection of all the λ -terms that can be constructed using Definition B.4.1. By convention, outermost parentheses are not written.

■ Example B.5

We use the letters x, y, z and variants with subscripts and primes to denote variables in V . To denote elements of Λ , we use L, M, N, P, Q, R and variants thereof. Syntactically identity of two λ -terms will be denoted by the symbol \equiv .

Definition B.4.2 The multiset of subterms, denoted Sub , is inductively defined as:

- i) (Variable) $Sub(x) = \{x\}$, for each $x \in V$.
- ii) (Application) $Sub((MN)) = Sub(M) \cup Sub(N) \cup \{(MN)\}$.
- iii) (Abstraction) $Sub((\lambda x. M)) = Sub(M) \cup \{(\lambda x. M)\}$.

We call L a subterm of M if $L \in Sub(M)$.

Lemma 3. (Reflexivity) For all λ -terms M , we have $M \in Sub(M)$. **(Transitivity)** If $L \in Sub(M)$ and $M \in Sub(N)$, then $L \in Sub(N)$.

■ Definition B.4.3 L is a proper subterm of M if L is a subterm of M , but $L \neq M$.

Parentheses in an outermost position may be omitted; application is left-associative; application takes precedence over abstraction; successive abstractions may be combined in a right-associative way under one λ .

If necessary, some parentheses should be added during the construction process.

Evaluation: Where an expression of the form $(\lambda x. M)N$ is rewritten to the expression $M[x := N]$ (the expression M in which every x has been replaced with N). We call this process β -reduction.

Reduction can be extended from subexpressions of bigger expressions (compatibility). The behavior of a function of two (or more) arguments can be simulated by converting it into a composite of a single argument (currying).

Types is an abstraction of the process of classifying entities into greater units.

B.5 Category Theory

C. Advanced Mathematics

*We are to admit no more causes of natural things
than such as are both true and sufficient
to explain their appearances.*

Isaac Newton

TODO: Provide a definition and some basic properties.

$$\log_a x = \frac{1}{\log_b a} \log_b x \quad (C.1)$$

TODO: And introduce the following inequation

$$\ln \frac{1}{x} \geq 1 - x \quad (C.2)$$

with the equality if, and only if, $x = 1$.

TODO: Characterization of balanced trees with logs.

TODO: The length of the canonical encoding of integers using logs.

C.1 Distinctiveness of Metrics

Let's \mathbf{X} be a dataset composed by m samples, y a response variable that can take two possible values 1 or -1 , \hat{f} a model trained using a supervised machine learning algorithm, and \hat{y} the vector of predicted values by \hat{f} when given as input \mathbf{X} (that is, $\hat{y} = \hat{f}(\mathbf{X})$). There exists multiple ways to measure the quality of the predicted values \hat{y} , like for example accuracy, precision, recall, F1 score, etc. In this technical report we are interested in compare the *distinctiveness* [REF], that is, the capacity to discriminate between multiple candidate \hat{y} , of the different error metrics. The closer to the theoretical limit of 2^m possible values, the better is the metric for model training purposes, since the risk of getting stuck in a local minima is smaller [REF].

A *confusion matrix* [REF] is a table that allows to visualize the performance of a trained model. Each row of the matrix represents the number of instances in a predicted class of \hat{y} while each column represents the number of instances in an actual class of y .

[Fix Table]

		Actual Class	
		Positive	Negative
Predicted Class	Positive	tp	fp
	Negative	fn	tn

We denote by p be the number of 1 in y , and by n the number of -1 (we have that $m = p + n$). The following figure depicts a graphical representation of these concepts. The left area of the square correspond to the positive training values, and the right to the negative. The dark gray area represent the values predicted as positive, and the light gray area the values predicted as negative.

(insert figure)

For an easier comparison between metrics we will simplify the distinctiveness of each metric using *big – O* (asymptotic) notation. Recall that a function $f(a)$ is of order $\mathcal{O}(g(a))$ if there exists a positive constant c such that $0 \leq f(a) \leq cg(a)$, for sufficiently large a .

C.2 Distinctiveness of Metrics

C.2.1 Accuracy

Accuracy is the ratio between the number of correct predictions and the total number of testing values.

Definition C.2.1 The accuracy of the predictions \hat{y} with respect to the training values y is given by:

$$acc = \frac{tp + tn}{tp + fp + tn + fn}$$

Proposition C.2.1 The total number of possible values for the metric accuracy is given by:

$$\#acc = m + 1$$

Proof. The denominator $tp + fp + tn + fn$ is fixed to the value m , meanwhile the numerator could go from no values correctly predicted to all values correctly predicted, and so, accuracy could take one of the following $m + 1$ possible values:

$$\frac{0}{m}, \frac{1}{m}, \dots, \frac{m}{m}$$

■

Corollary C.2.2 The order of total number of possible values for the metric accuracy is:

$$\mathcal{O}(m)$$

Proof. Given Proposition [prop:Accuracy].

■

C.2.2 Precision

Precision is the ratio between the correctly predicted positive values and the total number of predicted positive values.

Definition C.2.2 The precision of the predictions \hat{y} with respect to the testing values y is defined as:

$$pre = \frac{tp}{tp + fp}$$

Proposition C.2.3 The total number of possible values for the metric precision is given by:

$$\#pre \leq (p+1)(n+1)$$

Proof. tp can take $p+1$ different values (from no true positives to p true positives), and fp can take $n+1$ different values (from no false positive to n false positives), being the two values independent of each other. Precision can take at most $(p+1)(n+1)$ possible values, since some of these fractions can be simplified into a common distinctive value (how many fractions can be simplified depends on the function $\pi(q)$, that is, the number of primes less than q , which is unknown.) ■

Corollary C.2.4 The order of total number of possible values for the metric precision is:

$$\mathcal{O}(m^2)$$

Proof. Given Proposition [prop:Precision]. ■

C.2.3 Recall

Recall measures the ratio between the number of correctly predicted positive values and the total number of positives.

Definition C.2.3 The recall of the predictions \hat{y} with respect to the testing values y is defined as:

$$rec = \frac{tp}{tp + fn}$$

Proposition C.2.5 The total number of possible values for the metric recall is given by:

$$\#rec = (p+1)$$

Proof. Given that

$$rec = \frac{tp}{tp + fn} = \frac{tp}{p}$$

and that tp can take $p+1$ different values (from no true positives to all true positives). ■

Corollary C.2.6 The order of total number of possible values for the metric recall is:

$$\mathcal{O}(m)$$

Proof. Given Proposition [prop:Recall]. ■

C.2.4 F1

F_1 is the harmonic mean between precision and recall.

Definition C.2.4 The F_1 score of the predictions \hat{y} with respect to the testing values y is defined

as:

$$F_1 = \frac{2}{\frac{1}{pre} + \frac{1}{rec}}$$

Proposition C.2.7 The total number of possible values for the metric F1 is given by:

$$\#F1 \leq (n+1) \frac{(p+2)(p+1)}{2}$$

Proof. It is a well known property that

$$F_1 = \frac{2}{\frac{1}{pre} + \frac{1}{rec}} = \frac{2 \times pre \times rec}{pre + rec} = \frac{2tp}{2tp + fp + fn}$$

tp can take $p+1$ different values (from no true positives to p true positives), and fp can take $n+1$ different values (from no false positive to n false positives), being these two values independent. Fixed a tp the number of fn is $p - tp + 1$, so the total number of tp and fn is given by:

$$\sum_{tp=0}^p (p - tp + 1) = p(p+1) - \frac{p(p+1)}{2} + (p+1) = \frac{2p(p+1) - p(p+1) + 2(p+1)}{2} = \frac{(p+2)(p+1)}{2}$$

Then, F_1 can take at most $(n+1) \frac{(p+2)(p+1)}{2}$ possible values, since some of these fractions can be simplified into a common distinctive value. ■

Corollary C.2.8 Corollary 12. The order of total number of possible values for the metric F1 is:

$$\mathcal{O}(m^3)$$

Proof. Given Proposition [prop:F1]. ■

C.2.5 Area under the ROC curve

The area under the Receiver Operating Characteristic (AUC) measures how well our model differentiates between the distributions of the two classes.

Definition C.2.5 The area under the Receiver Operating Characteristic (AUC) score of the predictions $\hat{\mathbf{y}}$ with respect to the testing values \mathbf{y} is defined as:

$$F_1 = \frac{2}{\frac{1}{pre} + \frac{1}{rec}}$$

Proposition C.2.9 The total number of possible values for the metric F1 is given by:

$$\#F1 \leq (n+1) \frac{(p+2)(p+1)}{2}$$

Proof. The area under the Receiver Operating Characteristic (AUC) for the binary case can be computed as:

$$AUC = \frac{\sum_{i=1}^p (r_i - 1)}{pn}$$

The different values for the numerator $\sum_{i=1}^p (r_i - 1)$ correspond to the number of possible ways in which the p values could appear in the testing dataset, that is given by

$$\binom{m}{p} = \frac{m!}{n!p!} = \frac{m(m-1)\cdots(m-p+1)}{p(p-1)\cdots 1}.$$

The denominator is the constant $m!$. ■

Corollary C.2.10 Corollary 15. The order of total number of possible values for the metric AUC is:

$$\mathcal{O}(AUC) = m^{(m/2)}$$

Proof. Given Proposition [prop:AUC]. ■

Kolmogorov Inaccuracy

Kolmogorov inaccuracy is a measure of the distance between the predicted values and the real values.

Definition C.2.6 The Kolmogorov inaccuracy of the predictions $\hat{\mathbf{y}}$ with respect to the testing values \mathbf{y} is defined as:

$$kol = NCD(\mathbf{y}, \hat{\mathbf{y}})$$

Proposition C.2.11 The total number of possible values for the metric Kolmogorov inaccuracy is given by:

$$\#kol \leq \frac{(p+2)(p+1)(n+2)(n+1)}{2}$$

Proof. Proof. The normalized compression distance between two vectors \mathbf{y} and $\hat{\mathbf{y}}$ is given by

$$NCD(\mathbf{y}, \hat{\mathbf{y}}) = \frac{\hat{K}_C(\hat{\mathbf{y}}, \mathbf{y}) - \min\{\hat{K}_C(\hat{\mathbf{y}}), \hat{K}_C(\mathbf{y})\}}{\max\{\hat{K}_C(\hat{\mathbf{y}}), \hat{K}_C(\mathbf{y})\}}$$

using as compressor a code with optimal length, the Kolmogorov inaccuracy is given by:

$$\frac{-tp \log_2 \frac{tp}{m} - fp \log_2 \frac{fp}{m} - tn \log_2 \frac{tn}{m} - fn \log_2 \frac{fn}{m}}{\max\{-(tp+fp) \log_2 \frac{(tp+fp)}{m} - (tn+fn) \log_2 \frac{(tn+fn)}{m}, -p \log_2 \frac{p}{m} - n \log_2 \frac{n}{m}\}} - \frac{\min\{-(tp+fp) \log_2 \frac{(tp+fp)}{m} - (tn+fn) \log_2 \frac{(tn+fn)}{m}, -p \log_2 \frac{p}{m} - n \log_2 \frac{n}{m}\}}{\max\{-(tp+fp) \log_2 \frac{(tp+fp)}{m} - (tn+fn) \log_2 \frac{(tn+fn)}{m}, -p \log_2 \frac{p}{m} - n \log_2 \frac{n}{m}\}}$$

tp can take $p+1$ different values (from no true positives to p true positives), and tn can take $n+1$ different values (from no true negative to n true negatives), being this two values independent. Fixed a tp the number of fn is $p - tp + 1$, so the total number of tp and fn is given by:

$$\sum_{tp=0}^p (p - tp + 1) = p(p+1) - \frac{p(p+1)}{2} + (p+1) = \frac{2p(p+1) - p(p+1) + 2(p+1)}{2} = \frac{(p+2)(p+1)}{2}$$

and fixed a tn the number of fp is $n - tn + 1$, so the total number of tn and fp is given by:

$$\sum_{tn=0}^n (n - tn + 1) = n(n+1) - \frac{n(n+1)}{2} + (n+1) = \frac{2n(n+1) - n(n+1) + 2(n+1)}{2} = \frac{(n+2)(n+1)}{2}$$

Combining both expression, and taking into account that some of these fractions can be simplified into a common distinctive value, we get the desired result. ■

Corollary C.2.12 The order of total number of possible values for the metric Kolmogorov Accuracy is:

$$\mathcal{O}(m^4)$$

Proof. Given Proposition [prop:Accuracy]. ■

Conclusion

In this technical report we have derived exact formulas for the distinctiveness of the most common metrics used in binary classification algorithms, and the novel metric of Kolmogorov inaccuracy. The highest value of distinctiveness is achieved with the Kolmogorov inaccuracy, followed by the F1 metric. The lowest values are for accuracy and recall.

Perhaps, an easier way to compare the different metrics in terms of distinctiveness would be to use asymptotic notation. In this sense, accuracy and recall will have a distinctiveness of $\mathcal{O}(m)$, precision of $\mathcal{O}(m^2)$, F1 of $\mathcal{O}(m^3)$, and Kolmogorov inaccuracy of $\mathcal{O}(m^4)$. This result is in line with the intuition, since accuracy and recall depend on only one parameter, precision on two, F1 on three, and Kolmogorov inaccuracy on four.

D. Coq Proof Assistant

*We are to admit no more causes of natural things
than such as are both true and sufficient
to explain their appearances.*

Isaac Newton

TODO: Say something about Coq and the calculus of inductive constructions.
Warning! Axioms and propositions are not finished!

```
Require Import Arith.
```

Helper lemmas

Basic properties that are needed to prove some results.

```
Lemma less_or_equal (n m : nat):  
  n <= m <-> (n < m \vee n = m).  
Proof.  
  split.  
  intros.  
  induction H.  
  right.  
  reflexivity.  
  left.  
  assert (m < S m).  
  auto.  
  firstorder.  
  intros.  
  destruct H as [H1 | H2].  
  induction H1.  
  auto.  
  auto.  
  induction H2.
```

```
auto.
Qed.
```

Objects

Length function.

```
Definition l := fun n : nat => Nat.log2 n.
```

Oracle function, from descriptions to natural numbers.

```
Parameter O : nat -> nat.
```

Oracle equivalence relation, from pairs of descriptions to boolean.

```
Parameter R : nat -> nat -> Prop.
Axiom reflexive: forall r : nat, R r r.
Axiom symmetric: forall r s : nat, R r s -> R s r.
Axiom transitive: forall r s t : nat, R r s -> R s t -> R r t.
```

Nescience function, from descriptions to nescience.

```
Parameter N : nat -> nat.
```

Axioms

Axiom of non-negativity.

```
Axiom non_negativity: forall d : nat, N(d) >= 0.
```

Axiom of surfeit.

```
Axiom surfeit: forall s t : nat, R s t /\ 0 s <= 0 t /\ 1 s < 1 t -> N s < N t.
```

Axiom of inaccuracy.

```
Axiom inaccuracy: forall s t : nat, R s t /\ 0 s < 0 t /\ 1 s <= 1 t -> N s < N t.
```

Axiom of equality.

```
Axiom equality: forall s t : nat, R s t /\ 0 s = 0 t /\ 1 s = 1 t -> N s = N t.
```

Axiom of perfect knowledge.

```
Axiom perfect_knowledge: forall s : nat, N s = 0 <-> ( 0 s = 0 )
/\ ( ~ exists t : nat, s <> t /\ R s t /\ 0 t = 0 /\ 1 t < 1 s ).
```

Axiom of zero unknown.

```
Axiom zero_unknown: forall s : nat, exists t : nat, R s t /\ N t = 0.
```

Properties

If $N(d) = 0$ then $O(d) = 0$.

```

Lemma zero_inaccuracy (d : nat) :
  N d = 0 -> O d = 0.
Proof.
  intros.
  apply perfect_knowledge.
  assumption.
Qed.

```

If $l(s) < l(t)$ and $O(s) < O(t)$ then $N(s) < N(t)$.

```

Lemma property_ll:
  forall s t : nat, R s t -> l s < l t -> O s < O t -> N s < N t.
Proof.
  intros s t H1 H2 H3.
  apply surfeit.
  split.
  assumption.
  split.
  apply less_or_equal.
  left.
  apply H3.
  apply H2.
Qed.

```

If $l(s) < l(t)$ and $O(s) = O(t)$ then $N(s) < N(t)$.

```

Lemma property_le:
  forall s t : nat, R s t -> l s < l t -> O s = O t -> N s < N t.
Proof.
  intros s t H1 H2 H3.
  apply surfeit.
  split.
  assumption.
  split.
  apply less_or_equal.
  right.
  apply H3.
  apply H2.
Qed.

```

If $l(s) = l(t)$ and $O(s) < O(t)$ then $N(s) < N(t)$.

```

Lemma property_el:
  forall s t : nat, R s t -> l s = l t -> O s < O t -> N s < N t.
Proof.
  intros s t H1 H2 H3.
  apply inaccuracy.
  split.
  assumption.
  split.
  assumption.
  apply less_or_equal.
  right.
  apply H2.
Qed.

```

The property that if $l(s) = l(t)$ and $O(s) = O(t)$ then $N(s) = N(t)$ is given by the axiom of equality.

If $l(s) = l(t)$ and $O(s) > O(t)$ then $N(s) > N(t)$.

```

Lemma property_eg:
  forall s t : nat, R s t -> l s = l t -> O s > O t -> N s > N t.
Proof.
  intros s t H1 H2 H3.
  apply property_el.
  apply symmetric.
  assumption.
  auto.
  auto.
Qed.
```

If $l(s) > l(t)$ and $O(s) = O(t)$ then $N(s) > N(t)$.

```

Lemma property_ge:
  forall s t : nat, R s t -> l s > l t -> O s = O t -> N s > N t.
Proof.
  intros s t H1 H2 H3.
  apply property_le.
  apply symmetric.
  assumption.
  auto.
  auto.
Qed.
```

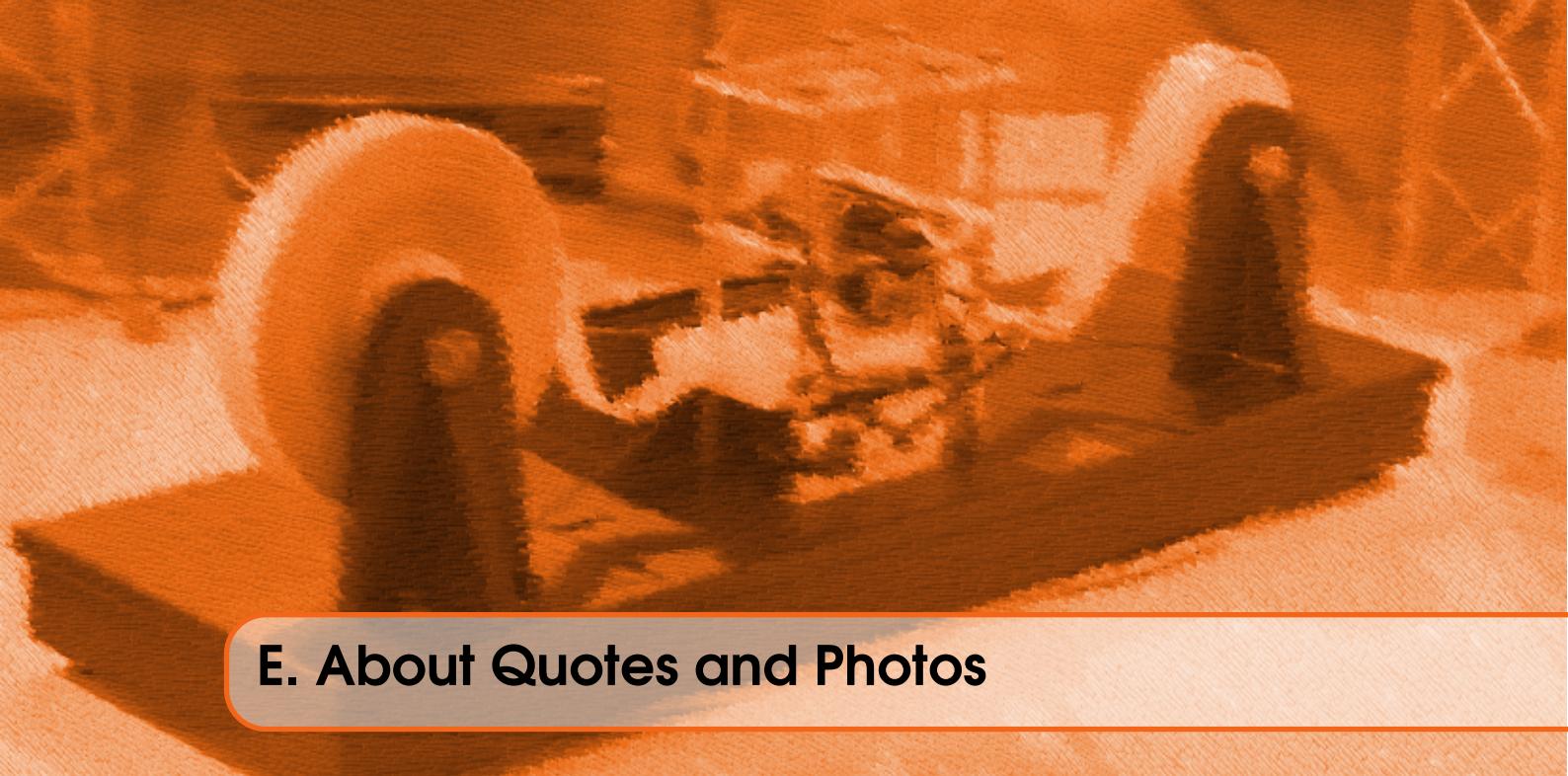
If $l(s) > l(t)$ and $O(s) > O(t)$ then $N(s) > N(t)$.

```

Lemma property_gg:
  forall s t : nat, R s t -> l s > l t -> O s > O t -> N s > N t.
Proof.
  intros s t H1 H2 H3.
  apply property_ll.
  apply symmetric.
  assumption.
  auto.
  auto.
Qed.
```

A conclusion cannot be derived from the following premises given the axioms.

- (i) If $l(s) < l(t)$ and $O(s) > O(t)$.
- (ii) If $l(s) > l(t)$ and $O(s) < O(t)$.



E. About Quotes and Photos

*What we know is little,
combined with tenacious concentration on a subject
and what we are ignorant of is immense.*
Pierre-Simon Laplace

TODO: Explain why are important quotes and photos

E.0.1 Quotes

I introduce this section.

Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away. Antoine de Saint-Exupéry.

This is a critical idea in the theory of nescience, although there are some differences. I think Saint-Exupéry is talking more about what we have called redundancy, rather than the concept of surfeit. Moreover, Saint-Exupéry is true as long as the inaccuracy is zero.

Computers are useless, they can only give you answers. Pablo Picasso.

As I have said in the Preface of the book, this quote triggered everything.

If presented with a choice between indifferent alternatives, then one ought to select the simplest one. Occam's razor principle.

I am sure that many people will claim that the theory of nescience is just Occam on steroids. By I think there are big differences, as I have already described in the book.

Mathematics may be defined as the subject in which we never know what we are talking about, nor whether what we are saying is true. Bertrand Russell.

Maybe I can talk here about Hilbert-Frege controversy, and what Russell means with this quote.

Sometimes it's the people no one imagines anything of who do the things that no one can imagine. Alan Turing.

Hopefully Turing is talking about me :)

Information is the resolution of uncertainty. Claude Shannon.

TODO: Explain

Some mathematical statements are true for no reason, they're true by accident. Gregory Chaitin.

TODO: Explain

All great work is the fruit of patience and perseverance, combined with tenacious concentration on a subject over a period of months or years. Santiago Ramón y Cajal.

Mention the book of Cajal.

To go where you don't know, you have to go the way you don't know. San Juan de la Cruz

TODO: Explain

We are all agreed that your theory is crazy. The question which divides us is whether it is crazy enough. Niels Bohr.

TODO: Explain

Wanderer, there is no road, the road is made by walking. Antonio Machado.

TODO: Explain

A little inaccuracy sometimes saves tons of explanations. Saki.

TODO: Explain

Everything should be made as simple as possible, but not simpler. Albert Einstein

TODO: Explain

There are known knowns. These are things we know that we know. There are known unknowns. That is to say, there are things that we know we don't know. But there are also unknown unknowns. There are things we don't know we don't know. Donald Rumsfeld.

TODO: Explain

It is not the answer that enlightens, but the question. Eugène Ionesco.

TODO: Explain

Invert, always invert. Carl Gustav Jacob Jacobi.

TODO: Explain

Always look for tricks. Antonio García.

TODO: Explain

Anyone who regards games simply as games and takes work too seriously has grasped little of either. Heinrich Heine.

TODO: Explain

Science may be regarded as the art of data compression. Li & Vitányi.

TODO: Explain

To be surprised, to wonder, is to begin to understand. José Ortega y Gasset.

TODO: Explain

We are to admit no more causes of natural things than such as are both true and sufficient to explain their appearances. Isaac Newton

TODO: Explain

What we know is little, and what we are ignorant of is immense. Pierre-Simon Laplace

TODO: Explain

When academics encounter a new idea that doesn't conform to their preconceptions, there's often a sequence of three reactions: first dismiss, then reject, and finally declare it obvious. S. Sloman and P. Fernbach.

TODO: Explain

E.0.2 Photos

In this Appendix we explain the intended meaning of the photographs included at the beginning of each chapter. All the photographs are royalty-free (or at least this is what Google Images says). Photographs have been pre-processed with GIMP¹, the GNU image manipulation program: we have applied a dotify filter (Artistic filter, GIMPressionist), and then altered the color map (Colors, Colorize) with a Hue/Saturation/Lightness levels of 24/84/10.

The Torch Bearers



The Torch Bearers is an aluminum sculpture created by the American artist Anny Hyatt Huntington, and donated to the city of Madrid in 1955. The sculpture is currently located at Universidad Complutense campus. The sculpture represents an old dying man that before to die pass the torch (a symbol of knowledge) to a young riding man that will continue the quest of perfect knowledge. The artist created other copies in bronze of the same sculpture that are located in Valencia, La Habana, and several cultural organizations around the United States.

Ancient Greek Philosophers



The carved busts of Greek philosophers Socrates, Antisthenes, Chrysippus, and Epicurus, located at the British Museum in London. The ideas and achievements of the ancient Greeks philosophers changed their world and had a huge influence in Western culture. Philosophers like Socrates, Plato and Aristotle formulated the first scientific explanations about how the world worked, and they pioneer a new way of thinking, based on reason and rational thought.

The scientific explanation of how the Universe works formulated by Aristotle was accepted as true during more than two thousands years.

Ars Magna

Ars Generalis Ultima or Ars Magna (The Ultimate General Art) was a book published by the Spanish philosopher Raimundo Lulio in 1305. The book contained the description of a mechanical device capable of answering any argument or question about Christian beliefs by means of using

¹www.gimp.org

logic and reason. The machine operated by rotating a collection of concentrically arranged circles to combine a fixed set of fundamental concepts. In this way, the device could show all possible truths about the subject of inquiry. The method was an early attempt to use logical means to produce new knowledge, and it was the inspiration for the methodology of finding interesting questions described in this book.

Galileo's Telescope



Galileo Galilei was an Italian astronomer, physicist, engineer, philosopher and mathematician. Galileo was the first scientist to clearly state the usefulness of mathematics to discover the laws of nature. Galileo also played a major role in the scientific revolution of the seventeenth century, introducing important innovations in the scientific method. In 1610, Galileo built a telescope and looked up at the heavens. His discoveries revolutionized the field

of astronomy and changed our understanding of the Universe.

Königsberg Bridges



The old city of Königsberg in Prussia (now Kaliningrad, Russia) was laid on both sides of the Pregel river. The city had two large islands which were connected to each other and to mainland by seven bridges. The seven bridges problem is a classical problem in mathematics that asks to devise a walk that would cross each of the seven bridges once and only once, starting and ending at any point, not necessarily the same. The Swiss mathematician Leonhard Euler

proved in 1736 that the problem has no solution. The work of Euler laid the foundations of a new mathematical discipline: Graph Theory.

The Turing Machine



In 1936, the British mathematician Alan Turing proposed a formal model for a hypothetical machine and claimed that his machine could compute anything that humans could compute following an algorithm. The model was a highly convincing one, and simple enough to allow precise mathematical analysis. In fact, the model had many of the ideas that ten years later electrical engineers used to build real computers. The Turing machine was one of this few moments in the history of science in which theory preceded practice.

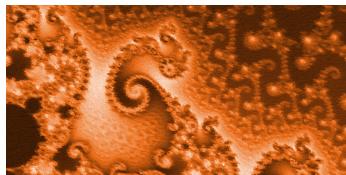
Morse key



A switching device used to send messages along a wire. The system sends pulses of electric current using the Morse code. Morse code is named after Samuel F. B. Morse, the inventor of the telegraph. The code is composed by a standardized sequence of short and long signals called *dots* and *dashes*, although it is not a binary code, since the code alphabet also includes symbols for the separation between letters and words. The average bit length per character

for the English language is 2.53, a remarkable result, given the fact that the code was designed intuitively, without knowing any of the (later discovered) results of coding theory.

Mandelbrot Set



The Mandelbrot set is created by sampling the complex numbers, and determining for each sample whether the result of iterating a function goes to infinity. Treating the real and imaginary parts of each number as image coordinates, pixels are colored according to how rapidly the sequence diverges, with black used for points where the sequence does not diverge. Images of the Mandelbrot set exhibit an elaborate boundary that reveals progressively ever-finer, self-similar, recursive detail at increasing magnifications. However, according to Kolmogorov complexity, the set presents a very low complexity.

XXX: XXX

XXX xxxx x xxx xx xxxxxx

Athena's Owl



A silver coin depicting the owl that traditionally accompanies Athena. Athena is the virgin goddess of wisdom in Greek mythology. The owl has been used as a symbol of knowledge, wisdom, perspicacity and erudition throughout the Western world, perhaps because their ability to see in the dark. The German philosopher Hegel famously noted that "the owl of [Athena] spreads its wings only with the falling of the dusk", meaning that philosophy comes

to understand a historical condition just as it passes away. In this sense, Hegel asserts that Philosophy cannot be prescriptive because it understands only in hindsight.

The Thinker



The Thinker is a bronze sculpture created by the French artist Auguste Rodin. The sculpture represents a nude male figure sitting on a rock with his chin resting on one hand. Originally created as part of a larger composition (The Gates of Hell), later the artist decided to treat the figure as an independent work, and at a larger size. There are about 28 full size castings located in museums and public places all around the world (Geneva, Brussels, San Francisco,

New York, Buenos Aires, etc.), and many others at different scales. A common interpretation of the sculpture is as an image of the deep thoughts required to find the right questions in philosophy.

R.U.R.



R.U.R. (Rossum Universal Robots) was a highly successful science fiction play written by the Czech Karel Čapek in 1920. The play introduced for the first time the word 'robot' as an alternative to other words used at that time like 'automaton' or 'android'. The word 'robot' derives from the Czech 'roboťa' meaning "forced labor of the kind that serfs had to perform on their masters' lands". The drama occurs in R.U.R., a factory that makes intelligent robots from artificial flesh and bones, so perfect that they can be mistaken for humans (the name Rossum is derived from the Czech word 'rozum' that means 'reason'). At the beginning robots were happy to work for humans, but then a rebellion starts and all the humans are murdered. At the end of the plot, robots realize that they do not have the knowledge to make new robots, and that by exterminating humans they have triggered their own extinction. The play is considered as a tragic satire about a naive humankind, the dangers of technology, and the obsolescence of God.

Wikipedia Monument



The Wikipedia Monument is located in the city of Słubice, Poland. The statue was designed by Armenian sculptor Mihran Hakobyan, and it was unveiled on October 2014, becoming the world's first monument to the online encyclopedia. The inscription reads: "With this monument the citizens of Słubice would like to pay homage to thousands of anonymous editors all over the world, who have contributed voluntarily to the creation of Wikipedia, the greatest

project co-created by people regardless of political, religious or cultural borders. In the year this monument is unveiled Wikipedia is available in more than 280 languages and contains about 30 million articles. The benefactors behind this monument feel certain that with Wikipedia as one of its pillars the knowledge society will be able to contribute to the sustainable development of our civilization, social justice and peace among nations."

F. Notation

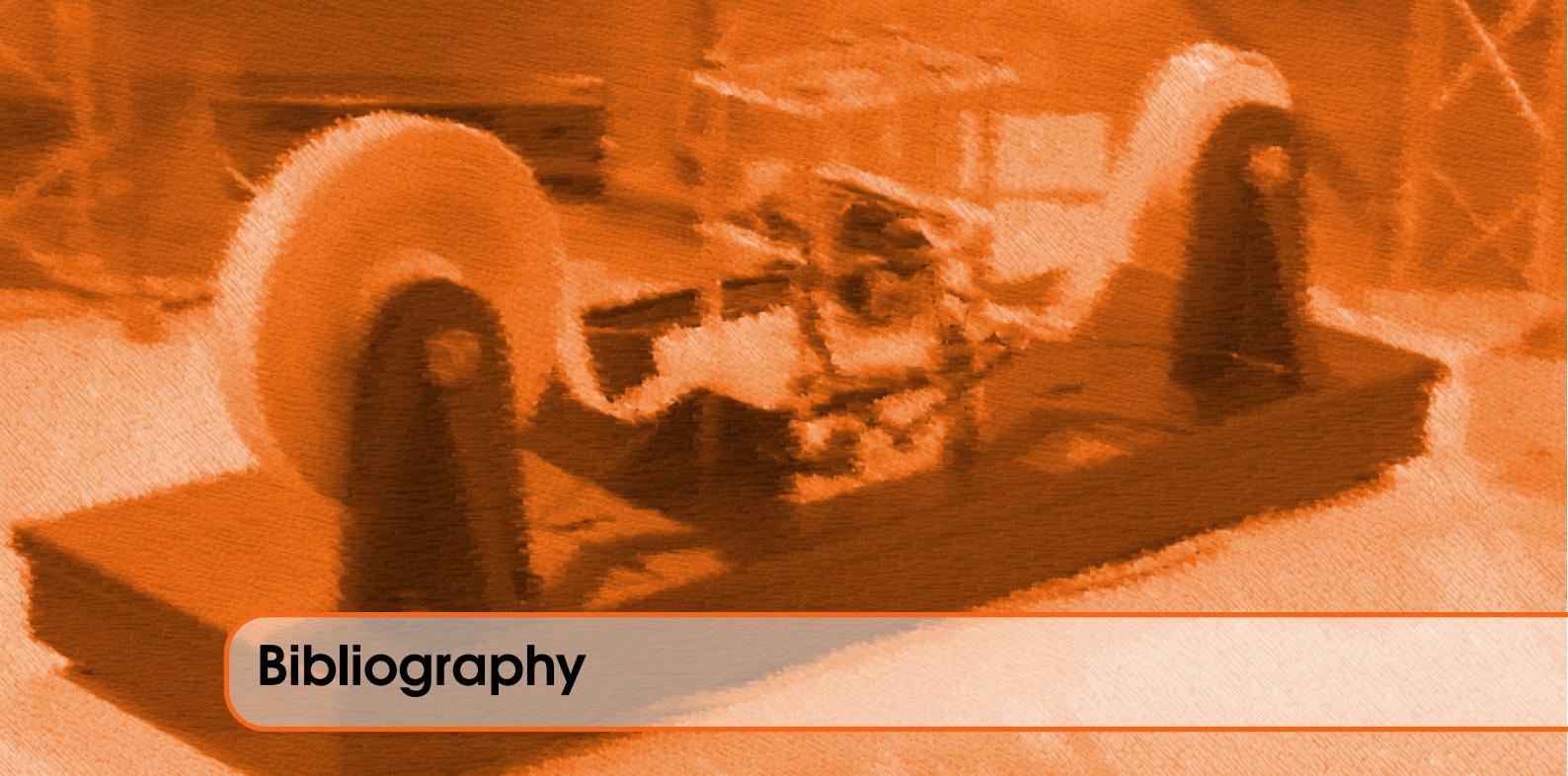
*When academics encounter a new idea that doesn't conform to their preconceptions,
there's often a sequence of three reactions:
first dismiss, then reject, and finally declare it obvious.*

S. Sloman and P. Fernbach

- \mathbb{N} set of natural numbers (including 0)
- \mathbb{Z} set or integers
- \mathbb{Z}^+ set or positive integers
- \mathbb{Q} set of rational numbers
- \mathbb{R} set of real numbers
- \mathbb{R}^+ set of positive real numbers
- $x \in A$ x is a member of A
- : set formation
- $A \subset B$ A is a subset of B
- $A \subseteq B$ A is a subset or equal to B
- \emptyset empty set
- $d(A)$ cardinality of A
- $A \cup B$ union of A and B
- $A \cap B$ intersection of A and B
- $A \setminus B$ set difference
- \bar{A} complement of A
- $\mathcal{P}(A)$ power set
- (x, y) ordered pair
- $A \times B$ cartesian product
- A^n n -fold cartesian product
- R binary relation
- \leq total order
- $\max(A)$ maximum

$\min(A)$	minimum
$f(x) = y$	function
$f(x) = \infty$	undefined element
I_A	identity
f^{-1}	inverse function
$f \circ g$	composition
1_A	characteristic function
$\text{abs}(x)$	absolute value
$\lceil x \rceil$	ceil
$\lfloor x \rfloor$	floor
$l(s)$	length of string
λ	empty string
s^R	reverse string
\mathcal{S}^n	set of strings of length n
\mathcal{S}^+	set of all finite strings
\mathcal{S}^*	set of all finite strings including the empty string
$<_p$	prefix
\bar{s}	Self delimited string
$\langle O \rangle$	Encoding as string of O
$G = (V, E)$	graph
$\deg(v)$	degree of a vertex
$N(v)$	neighborhood of a vertex
$\text{indeg}(v)$	in-degree of a vertex
$\text{outdeg}(v)$	out-degree of a vertex
Ω	sample space
$P(x)$	probability of x
$E(X)$	expectation of X
T	Turing machine
Q	set of states
Γ	set of tape symbols
\sqcup	blank symbol
Σ	input symbols
q_o	initial state
q_f	final state
τ	transition function
C	configuration
C_o	initial configuration
C_f	final configuration
U	universal Turing machine
\mathcal{E}	Set of entities
\mathcal{R}	Set of representations
$\mathcal{R}_{\mathcal{E}}$	Set of representations of \mathcal{E}
$t \in \mathcal{T}$	Research topic
$d \in \mathcal{D}_t$	Description of a topic
\mathcal{D}	Set of descriptions
$\mathcal{D}_{\mathcal{T}}$	Set of valid descriptions of \mathcal{T}
\mathcal{D}_t	Set of descriptions of $t \in \mathcal{T}$
δ	Description function

d_t^*	Perfect description of $t \in \mathcal{T}$
$d_{t,s}$	Joint description of $t, s \in \mathcal{T}$
$\mathcal{D}_{t,s}$	Set of joint descriptions of $t, s \in \mathcal{T}$
$d_{t,s}^*$	Perfect joint description of $t, s \in \mathcal{T}$
$d_{t s^*}$	Conditional description of t given $s, t, s \in \mathcal{T}$
$\mathcal{D}_{t s^*}$	Set of conditional descriptions of t given $s, t, s \in \mathcal{T}$
$d_{t s^*}^*$	Perfect conditional description of t given $s, t, s \in \mathcal{T}$
$A \subset \mathcal{T}$	Research area
\hat{A}	Know subset of the area $A \subset \mathcal{T}$
$\mathcal{D}_{\hat{A}}$	Description of the area A given the known subset \hat{A}
$\mathcal{D}_{\hat{A}}$	Set of descriptions of the area A given the known subset \hat{A}
$d_{\hat{A}}^*$	Perfect description of the area A given the known subset \hat{A}
RG	relevance graph
R_t	relevance of topic t
IP_t	interestingness of topic t as a problem
M_t	maturity of topic t
AG	applicability graph
A_t	applicability of topic t
IT_t	interestingness of topic t as a tool
T'	set of known topics
$Q_{t_1 \rightarrow t_2}$	interesting question
$IQ_{t_1 \rightarrow t_2}$	interestingness of question $Q_{t_1 \rightarrow t_2}$
\mathbb{F}	unknown frontier
\mathbb{S}	new topics area
$S_{\{t_1, t_2\}}$	new topic
$IS_{\{t_1, t_2\}}$	interestingness of a new topic
IT_A	interestingness of an area as tool
IP_A	interestingness of an area as problem
$\hat{i}(\hat{\mathbf{y}}, \mathbf{y})$	inaccuracy of predicted values



Bibliography

Books

Articles

