

# **The Mathematics of the Unknown**

Bringing to the present the knowledge of the future

**R. A. García Leiva**

(This book is 76% complete)

Copyright © 2022 R. A. García Leiva  
[www.mathematicsunknonw.com](http://www.mathematicsunknonw.com)

PUBLISHED BY THE AUTHOR



This book is dedicated to my wife Justi,  
my son Daniel, and my two daughters Teresa and Lucía.



# Contents

<b>1</b>	<b>Introduction</b>	<b>19</b>
1.1	Entities	20
1.2	Representations	21
1.3	Descriptions	22
1.4	Miscoding	24
1.5	Inaccuracy	25
1.6	Surfeit	27
1.7	Nescience	28
1.8	Evolution of Nescience	31
1.9	Other Metrics	32
1.10	Interesting Research Questions	35
1.11	New Research Entities	36
1.12	References and Further Reading	37

I

## Part 1: Background

<b>2</b>	<b>Discrete Mathematics</b>	<b>41</b>
2.1	Sets, Relations and Functions	42
2.2	Strings	44
2.3	Matrices	45
2.4	Graphs	45
2.5	Counting Methods	47

<b>3</b>	<b>Discrete Probability .....</b>	<b>49</b>
3.1	Foundations	50
3.2	Conditional Probability	53
3.3	Random Variables	55
3.3.1	Multivariate Distributions .....	57
3.3.2	Marginal Distribution .....	58
3.3.3	Conditional Distributions .....	58
3.4	Characterizing Distributions	59
3.4.1	Measures of Central Tendency .....	60
3.4.2	Measures of Dispersion .....	61
3.4.3	Measures of Statistical Relationship .....	62
3.5	Common Distributions	63
3.5.1	Uniform Distribution .....	63
3.5.2	Bernoulli Distributions .....	63
3.5.3	Binomial Distributions .....	64
3.6	Large Random Samples	64
3.6.1	Law of Large Numbers .....	64
3.6.2	Central Limit Theorem .....	66
<b>4</b>	<b>Computability .....</b>	<b>67</b>
4.1	Turing Machines	68
4.2	Universal Turing Machines	71
4.3	Non-Computable Problems	72
4.4	Computable Functions and Sets	73
4.5	Oracle Turing Machine	74
4.6	Computational Complexity	74
<b>5</b>	<b>Coding .....</b>	<b>77</b>
5.1	Coding	78
5.2	Kraft Inequality	80
5.3	Entropy	82
5.4	Optimal Codes	85
5.5	Huffman Algorithm	86
5.6	Discretization Algorithms	88
5.6.1	k-means Clustering .....	89
<b>6</b>	<b>Complexity .....</b>	<b>91</b>
6.1	Strings Complexity	92
6.2	Properties of Complexity	94
6.3	Conditional Kolmogorov complexity	95
6.4	Information Distance	96
6.5	Incompressibility and Randomness	97

<b>7</b>	<b>Learning .....</b>	<b>99</b>
<b>7.1</b>	<b>Statistical Inference</b>	<b>99</b>
7.1.1	Bayesian Inference .....	101
7.1.2	Non-Bayesian Inference .....	102
<b>7.2</b>	<b>Machine Learning</b>	<b>103</b>
7.2.1	Model Accuracy .....	104
7.2.2	No free lunch theorem .....	105
7.2.3	The bias-variance trade-off .....	105
7.2.4	Generative vs. discriminative models .....	105
7.2.5	Decision Trees .....	105
7.2.6	Time Series Analysis .....	106
<b>7.3</b>	<b>Minimum Message Length</b>	<b>110</b>
<b>7.4</b>	<b>Minimum Description Length</b>	<b>112</b>
7.4.1	Refined MDL .....	113
<b>7.5</b>	<b>Multiobjective Optimization</b>	<b>114</b>
7.5.1	Trade-offs .....	114
7.5.2	Optimization Methods .....	115
<b>8</b>	<b>Philosophy of Science .....</b>	<b>117</b>
<b>8.1</b>	<b>Metaphysics</b>	<b>117</b>
<b>8.2</b>	<b>Scientific Representation</b>	<b>118</b>
<b>8.3</b>	<b>Models in Science</b>	<b>119</b>
<b>8.4</b>	<b>Scientific Theories</b>	<b>120</b>
<b>8.5</b>	<b>The Scientific Method</b>	<b>120</b>
<b>8.6</b>	<b>Scientific Discovery</b>	<b>121</b>

## II

## Part 2: Foundations

<b>9</b>	<b>Entities, Representations and Descriptions .....</b>	<b>125</b>
<b>9.1</b>	<b>Entities</b>	<b>126</b>
<b>9.2</b>	<b>Representations</b>	<b>127</b>
<b>9.3</b>	<b>Joint Representations</b>	<b>130</b>
<b>9.4</b>	<b>Descriptions</b>	<b>132</b>
<b>9.5</b>	<b>Descriptions for Joint Representations</b>	<b>135</b>
<b>9.6</b>	<b>Conditional Descriptions</b>	<b>136</b>
<b>9.7</b>	<b>Research Areas</b>	<b>138</b>
<b>9.8</b>	<b>References</b>	<b>139</b>
<b>10</b>	<b>Miscoding .....</b>	<b>141</b>
<b>10.1</b>	<b>Miscoding</b>	<b>142</b>
<b>10.2</b>	<b>Miscoding of Joint Representations</b>	<b>144</b>
<b>10.3</b>	<b>Reducing Miscoding</b>	<b>145</b>
<b>10.4</b>	<b>Miscoding of Areas</b>	<b>145</b>

<b>11</b>	<b>Inaccuracy</b>	<b>147</b>
11.1	Inaccuracy	148
11.2	Conditional Inaccuracy	149
11.3	Inaccuracy of Areas	150
<b>12</b>	<b>Surfeit</b>	<b>151</b>
12.1	Surfeit	152
12.2	Joint Surfeit	153
12.3	Conditional Surfeit	155
12.4	Surfeit of Areas	156
<b>13</b>	<b>Nescience</b>	<b>159</b>
13.1	Nescience	160
13.2	Joint Nescience	162
13.3	Conditional Nescience	162
13.4	Nescience of Areas	163
13.5	Perfect Knowledge	164
13.6	Current Best Description	165
13.7	Nescience based on Datasets	166
13.8	Unknonwn Unknown	166
<b>14</b>	<b>Interesting Questions</b>	<b>167</b>
14.0.1	Combination of Topics	168
14.1	Relevance	169
14.2	Applicability	170
14.3	Interesting Questions	171
14.4	New Research Topics	172
14.5	Classification of Research Areas	174
<b>15</b>	<b>Advanced Properties</b>	<b>177</b>
15.1	The Axioms of Science	177
15.1.1	Model Theory	178
15.1.2	Type Theory	181
15.1.3	Category Theory	188
15.2	Scientific Method	188
15.2.1	Science as a Language	188
15.3	The Inaccuracy - Surfeit Trade-off	189
15.4	Science vs. Pseudoscience	189
15.5	Graspness	190
15.6	Effort	190
15.7	Human Understanding	190
15.8	Areas in Decay	191

<b>16</b>	<b>Machine Learning .....</b>	<b>195</b>
16.1	Nescience Python Library	195
16.2	A Note About Compression	196
16.3	Miscoding	198
16.4	Inaccuracy	205
16.5	Surfeit	208
16.6	Nescience	211
16.7	Auto Machine Classification	215
16.7.1	Surfeit of Algorithms .....	216
16.8	Auto Machine Regression	216
16.9	Time Series	216
16.9.1	Automiscoding, Crossmiscoding and Partial Automiscoding .....	216
16.9.2	Auto Time Series .....	219
16.10	Anomaly Detection	220
16.11	Decision Trees	222
16.11.1	Algorithm Description .....	223
16.11.2	Algorithm Evaluation .....	226
16.12	Algebraic Model Selection	230
16.13	The Analysis of the Incompressible	233
<b>17</b>	<b>Software Engineering .....</b>	<b>235</b>
17.1	Redundancy of Software	235
17.2	Quality Assurance	239
17.3	Forex Trading Robots	239
<b>18</b>	<b>Philosophy of Science .....</b>	<b>243</b>
18.1	Wikipedia, The Free Encyclopedia	243
18.2	Classification of Research Topics	243
18.3	Classification of Research Areas	247
18.4	Interesting Research Questions	248
18.5	Interesting Research Topics	249
18.6	Philosophy of Science	252
18.7	Evolution of Knowledge	254
18.8	Graspness of Topics	256
18.9	Probability of Being True	256
18.10	Unused text	257
<b>19</b>	<b>Computational Creativity .....</b>	<b>259</b>
19.1	Classification of Research Topics	260
19.1.1	Relevance .....	260

19.1.2	Applicability . . . . .	261
19.1.3	Maturity . . . . .	262
<b>19.2</b>	<b>Interesting Research Questions</b>	<b>262</b>
19.2.1	Intradisciplinary Questions . . . . .	262
19.2.2	Interdisciplinary Questions . . . . .	262
<b>19.3</b>	<b>New Research Topics</b>	<b>262</b>
<b>19.4</b>	<b>Classification of Research Areas</b>	<b>262</b>
<b>19.5</b>	<b>References</b>	<b>262</b>
<b>19.6</b>	<b>Future Work</b>	<b>263</b>

## IV

## Appendix

<b>A</b>	<b>Foundations of Mathematics</b> . . . . .	<b>267</b>
A.1	Propositional Logic	267
A.2	Predicate Logic	270
A.3	Set Theory	271
A.4	Lambda Calculus	272
A.5	Category Theory	273
<b>B</b>	<b>Advanced Mathematics</b> . . . . .	<b>275</b>
B.1	Distinctiveness of Metrics	275
B.2	Distinctiveness of Metrics	276
B.2.1	Accuracy . . . . .	276
B.2.2	Precision . . . . .	276
B.2.3	Recall . . . . .	277
B.2.4	F1 . . . . .	277
B.2.5	Area under the ROC curve . . . . .	278
<b>C</b>	<b>Coq Proof Assistant</b> . . . . .	<b>281</b>
<b>D</b>	<b>About Quotes and Photos</b> . . . . .	<b>285</b>
D.0.1	Quotes . . . . .	285
D.0.2	Photos . . . . .	287
<b>E</b>	<b>Notation</b> . . . . .	<b>291</b>
	<b>Bibliography</b> . . . . .	<b>295</b>
	<b>Books</b>	<b>295</b>
	<b>Articles</b>	<b>296</b>
	<b>Index</b> . . . . .	<b>299</b>



## 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 main idea of this book is that perfect knowledge implies randomness. This is, in principle, a highly counterintuitive idea, since a lot of effort in science deals with the task to name, organize and classify our messy, chaotic, world. Even the kind of knowledge that explains how things work requires a previous ordering and classification. Science, apparently, is anything but random. Yet, this is not the case.

If it requires a lot of time to explain how something works, probably we do not fully understand it. For example, consider the calculus of derivatives. At its origin, during the time of Newton and Leibniz, it was required a lot of space to define the concept of limit of a function. Moreover, only the specialists, if any, were able to understand the idea. We had a highly incomplete knowledge. Today, the definition of derivative barely takes one paragraph in a math book, and it is taught at high school. Our current knowledge is much better. Long explanations are usually superfluous, they contain repeated ideas, unused concepts, improperly identified relations, bad notation, and so on. With a better understanding, normally after considerable research, we should be able to remove those unnecessary elements. And when there is nothing left to take away, we say that we have achieved a perfect knowledge. Hence, if a theory is perfect, that is, it presents no redundant elements we can remove, its description must be an incompressible string. And that is exactly the mathematical definition of random string, an incompressible sequence of symbols. In this sense, a scientific theory is random if it contains the maximum amount of information in the less space possible.

In this book it is described the new *Theory of Nescience*, a mathematical theory that address the problem of what science is and how scientific knowledge is acquired. Our main assumption is that it is easier to measure how much we do not know than measuring how much we do know, because randomness posses a limit to how much we can know. A second assumption is that the computer, as a conceptual model, is the right tool to provide this quantitative measure.

The fact that randomness imposes a limit on how much we can know about a particular research topic, far from being a handicap, opens new opportunities in science and technology. The proper understanding of this limit will allow us not only to solve the most challenging open problems, but also to discover new interesting research questions. In the book I also describe some of the practical applications of the theory of nescience to the areas of scientific research, artificial intelligence, computational creativity and software engineering.

## **Research Agenda**

The theory of nescience has been developed with the aim to provide answers to the following questions:

*Q1: Can we provide a quantitative characterization of how much we do not know about a research topic?* Given this metric we could not only measure how much we do not know about a particular topic, for example climate change, but also we could measure how much a new development or idea (usually published in the form of a research paper) contributes to increase our knowledge. If we combine this metric with a measure of the relevance of the problem investigated, we could quantify the value of new scientific contributions.

*Q2: Is it possible to compare how much we do not know about topics from unrelated scientific areas?* If so, we could classify and compare all the open questions according to how much we do not know about them, regardless of the disciplines to which they belong (physics, biology, sociology, etc.). This classification, together with the already mentioned metric of relevance, will allow us to decide where we should concentrate our research efforts. Of course, I am not proposing to stop doing research in basic or fundamental science, on the contrary, I will show we need this kind of science.

*Q3: How can we distinguish between science and pseudo-science?* This is a long standing, still open, question in philosophy of science. It would be very nice to have a mathematical answer that can be implemented in practice, since that would allow us to assess how scientific are some controversial disciplines that call themselves science.

*Q4: Are some research topics inherently more difficult to understand than others?* So we could elucidate if this true that researchers of some disciplines are smarter than researchers of other disciplines as they claim, or it is just a matter that the former are far easier to understand than the laters.

*Q5: Are there topics beyond the capabilities of human comprehension?* What are the limits of human understanding? Maybe there exists problems that can not be solved by humans, and research topics that our limited brain can not comprehend. Perhaps we should get used to that progress, in certain research areas, can be only achieved if we give-up and let computers do the creative scientific work.

*Q6: Is there a systematic procedure to increase our knowledge?* It might happen that what we call the "scientific method" is in fact a non-solvable problem, and the only thing we can provide is a practical approximation. And, if this is the case, how do we evaluate and compare different approximations? Can we provide a new and better method based on the principles of the theory of nescience?

*Q7: What is the effort required to fully understand an unknown topic?* For example, how much does it cost to increase by 1% our understanding of how to cure cancer? Given this metric we could plan our research activities according to priorities, budget, and relevance of potential results. Of course, this metric would be a lower bound, since we could still do it much worse, for example, due to bad project management.

*Q8: What is perfect knowledge? Can perfect knowledge be achieved for all possible entities?* Randomness is a necessary condition for perfect knowledge, but we need something more to characterize what full understanding means. Moreover, we have to prove if perfect knowledge is

always achievable or, on the contrary, there are fundamental limitations that make perfection in science a chimera.

*Q9: Is there a procedure to discover new, previously unknown, interesting research entities and problems?* We need a method to explore the unknown unknown, that is, those problems that not only we do not know how to solve, but also, we are not even aware they exists. Such a method would allow us to bring to the present the research topics of the future.

Some of the answers provided in this book are more mature than others. For some questions I will provide a full theoretical answer with a practical implementation, and for others, just a rough sketch of what a solution could be. But I am firmly convinced that with more research, the new theory of nescience can provide satisfactory answers to all the posed questions. In this sense, this book is more a research agenda than a comprehensive description of a full theory of nescience.

### Origins of the Theory of Nescience

I think it was in 1991 (when I was eighteen) that I read for the first time the sentence "*Computers are useless, they can only give you answers*". The quote is attributed to Pablo Picasso, one of the most creative and influential artists of the 20th century. Soon I realized how terrible right he was, since it is true that computers cannot make interesting, original, questions. But it was more than 20 years later, in 2014, when I started to look seriously to the challenge that Picasso posed to the computer science community.

Most of the concepts behind my methodology for the discovery of interesting questions were formulated in just one night that I could not sleep: nescience, relevance, unknown unknown, and many others. Or perhaps they were developed by my subconscious mind during all those 20 years, since I do not think it was a matter of luck that I decided to pursue subjects like information theory or Kolmogorov complexity long before I was aware that they were the foundation of a future theory of nescience. What it is remarkable is that it took me just one single night to come up with a rough sketch of the most important ideas, a couple of months to develop the initial computer experiments that validated them, and several years to fully develop the mathematics needed to provide the theoretical support.

At the beginning, I was mostly interested in finding interesting scientific questions, that is, to discover what it is hidden in what I call the unknown unknown area, with the aim of bringing to the present the research topics of the future. Later on I started to look into the evolution of the nescience of scientific topics over time, and I asked myself what would happen if we keep improving a theory forever. It is then when I realized that perfect knowledge implies randomness, and the original methodology for the discovery of interesting questions was extended into a full theory of nescience to develop this important idea.

I may confess that the new theory has had, in general, a pleasant welcome form my colleges and those scientists to whom I had the opportunity to talk during the initial development stages. This early success encouraged me to continue with a further improvement of the main ideas and their applications in practice. Meanwhile I was developing the mathematics supporting the concept of nescience, I started to look into other potential applications, including the analysis of computer programs (quality assurance) and raw datasets (machine learning).

**TODO: Remove the following paragraph if at the end these results are not included in this version of the book.**

However, I was not satisfied with the status of the theory in spite of its explanatory power. It is true that given the theory I was able to provide solutions to some open questions in the area of philosophy of science, for example, how well we understand mathematics compared to sociology, why some research topics are more difficult than others, what it is the fundamental difference between science and pseudoscience, and many others. But the problem was that the theory did not provide any predictions that could be falsified by means of running an experiment. So I decided to

take the risk and further develop the mathematics to provide such kind of predictions. This is when I came up with the function that describes how nescience decreases as we increase our research effort. This function allows us to predict the maximum increment of knowledge that we can gather about a topic given the amount of effort we are willing to put in its understanding. But, still not fully satisfied, I decided to push the theory one step forward and try to find a novel prediction, that is, something that had not been observed before. New developments of the mathematical foundations behind the theory allowed me to discover the highly counterintuitive property that sometimes, for a certain class of topics, further research could be a counterproductive activity. That is, that more research we do, the more confused we get, since for these topics it is not possible to increase our knowledge beyond a critical point, even if this point is far from a perfect knowledge.

### About the Book

The theory of nescience borrows concepts from multiple academic disciplines: computability, randomness, information theory, complexity, probability, graph theory, philosophy of science, and many more. However, the book is self-contained, and so, no previous knowledge is required to follow the mathematical developments, beyond familiarity with first year calculus, abstract algebra and some computer programming experience. The contents have been selected to cover the needs of readers with very different backgrounds: mathematicians, computer scientists, engineers, ... The math level is that of a graduate student or advanced undergraduate.

**TODO: Review and update once the book is finished.**

The book has been structured into three main parts: Background, Foundations and Applications. Readers already familiar with the mathematics covered by the Background section can directly proceed to the Foundations part. However, it is highly recommended to perform at least a quick review of the notation used in these chapters. After getting acquainted with the details of the theory of nescience described in the Foundations section, the reader can continue with the applications. It is not necessary to fully understand all the details of the theory in order to follow the applications. A basic knowledge of the main concepts and results should be sufficient.

- *Chapter 1 Introduction* contains a gentle introduction to the theory of nescience, and a short review of the main results. The chapter avoids the use of advanced mathematics, but the concepts are still semi-formally introduced. Although it is not recommended, those readers that cannot follow the mathematics behind the theory could just read this first chapter, the introductory sections of the remaining chapters of the Background and Foundations parts, and then proceed directly to the applications.

*PART I Background* is an introduction to the mathematics required to measure how much we do not know about a topic and how random is a string. The goal of this part is to fix notation, formally define concepts, and prove some important results. I do not assume any previous knowledge to follow the covered material, however, to fully understand these topics I recommend to refer to the standard literature (see the References section included at the end of each chapter). Additional topics are covered in the appendices.

- *Chapter 2 Discrete Mathematics* is an overview of the basic mathematics required to understand the more advanced topics covered in the book. This chapter is intended as a quick review of these concepts, and so, no formal definitions are provided and theorems are not proved. Among others, the topics covered include sets, relations, strings, graphs and discrete probability.
- *Chapter ??iscrete Probability* TODO.
- *Chapter 4 Computability* provides a formal definition of the concept of algorithm. We will introduce the concept of universal Turing machine, and we will see that there are well defined mathematical problems that cannot be solved by computers. The very important tools of

oracle Turing machine and Turing reducibility are also studied in detail. Computational complexity main results are briefly reviewed.

- In *Chapter 5 Coding* we will study the properties of codes, and in particular, how codes allow us to compress a text without loosing any information by means of removing statistical redundancy. We will see that there exists a limit to how much a text can be compressed by using this technique, and that this limit is given by the entropy of the source.
- *Chapter 6 Complexity* introduces an absolute metric, called Kolmogorov complexity, to measure the amount of information contained in a string, by computing the length of the shortest computer program that can print that string. The properties of this metric are studied in detail. The relation between string complexity and randomness is also discussed.
- *Chapter 7 Learning* studies the relation between codes and probabilities. It provides a short review of the area of statistical learning, with a focus on existing approaches that apply the concept of minimum string length to the problem of stochastic models evaluation and optimal parameters selection. An introduction to the concepts and notation of nonlinear multiobjective optimization problems is included as well.
- *Chapter 8 Philosophy of Science* is a short introduction to the discipline of philosophy of science. We will review concepts like scientific representations, models, theories and many other components of science from a philosophical point of view, identifying the most important elements that any formal theory of science should include. The chapter also contains a review of the current state of the art of what it is the scientific method.

*PART II Foundations* contains a detailed description of the theory of nescience, defining formally the concepts involved and proving the main theoretical results. This is the most important part of the book, where the new theory is fully developed. Readers with a solid background in computability, complexity, information theory and probability could proceed directly to this part.

- *Chapter 9 Entities, Representations and Descriptions* introduces the basic constituents of the theory of nescience: entities, representations and descriptions. The properties of these elements and how they interlace together are studied. We will see how multiple representations and descriptions can be combined, and how to include background knowledge in a research. The relation between perfect knowledge and randomness is discussed, and a novel concept of research area is proposed.
- *Chapter 10 Miscoding* address the difficult problem of how to represent abstract, and non-abstract, research entities as string of symbols that can be used for research. In the chapter we will formally introduce the concept of miscoding and study its properties. Miscoding is a quantity that measures the error introduced due to improper encodings.
- *Chapter 11 Inaccuracy* contains a novel interpretation of the classical concept of error, that is, how accurately a description explains an entity. The concept of inaccuracy is extended to deal with all kinds of topics, including abstract ones. The joint and conditional versions of the concept of inaccuracy are introduced and their properties studied.
- *Chapter 12 Surfeit* studies how redundant is a description, that is, how many unnecessary elements a description contains. Surfeit can be seen as a measure of how well we currently understand research topics and research areas. The joint and conditional versions of surfeit are provided as well.
- *Chapter 13 Nescience* is the central chapter of the book, since it contains the mathematical foundations of the new theory. In the chapter is formally defined the concept of nescience and its main properties studied, like for example, how nescience evolves with time, what we mean by perfect knowledge, and how to identified our current best model.
- *Chapter 14 Interesting Questions* describes a procedure to find new interesting things in a large collection of measurable objects, and in particular, to find new interesting research questions and new research topics. New concepts like relevance and applicability are defined

and studied.

- *Chapter 15 Advanced Properties* covers some advanced concepts and properties of the theory of nescience. These properties are not needed to understand the applications of the theory, and so, they can be safely skipped in a first read of the book (but, definitely, not in a second). Among others, new concepts introduced include: an axiomatic version of the theory, graspness as a measure of how difficult is a particular research topic, or the minimum effort required to reduce the nescience of a topic.

*PART III Applications* provides a collection of practical applications of the concept of nescience, in the areas of machine learning, software engineering, philosophy of science, and computational creativity. The applications have been selected to cover the entire spectrum of possible kinds of topics, from abstract ones (research topics), real objects and strings (computer programs).

- *Chapter 16 Machine Learning* describes how the concept of nescience can be applied in the case of entities represented by large datasets. In particular, the methodology will be applied to select relevant features, identify optimal models, and compute errors. Some new machine learning algorithms, like a novel approach to derive optimal decision trees, will be proposed as well.
- *Chapter 17 Software Engineering* covers how to use the theory of nescience with computer programs, in order to provide a measure of how well we understand current software platforms (operating systems, networking middleware, productivity tools, ...). We will evaluate if current versions of software are better than past versions, or if software is degrading with time. The results are also applied to the automatic discovery of errors, with the aim to improve the quality of software.
- *Chapter 18 Philosophy of Science* studies how to apply the new metrics introduced in this book to study of what it is science and how science is made. Some important open problems in the area of philosophy of science will be addressed, including the demarcation problem (how to distinguish between science from pseudoscience) and the question of how science make progress. Multiple proposals of what it is the scientific method are evaluated in the context of the theory of nescience, and compared against our own proposal.
- *Chapter 19 Computational Creativity* shows how to apply in practice the methodology for the discovery of interesting things. In particular, the methodology will be applied to find new research questions and new research topics, where multiple examples of interesting questions and topics will be provided.

*Appendix A Foundations of Mathematics* describes three different theoretical frameworks from which mathematics can be formalized: logic and set theory, dependent type theory, and category theory. These frameworks can also be used to provide a solid foundation to the theory of nescience.

*Appendix B Advanced Mathematics* contains a brief description of some mathematical concepts that have been used in the book, but that play a secondary role in the development of the theory; they are included in this appendix for reference.

*Appendix C Mechanical Proofs* provides a mechanical implementation of the axioms of the theory of nescience, and its most relevant results, based on the calculus of inductive constructions, and in particular, in the Coq proof assistant.

Last, but not least, *Appendix D About the Photos* explains the origin and intended meaning of the carefully selected photographs included at the header of each chapter.

## Acknowledgements

I would like to give thanks to all the people who has contributed with comments and ideas to the development of the theory of nescience. In particular, I would like to give thanks to Antonio

Fernández, Vincenzo Mancuso and Paolo Casari who believed and supported this project since its very beginning, when it was only a crazy nonsense idea (although perhaps it still is). Other people who has provided contributions and useful comments where Héctor Cordobés, Luis F. Chiroque, Agustín Santos, Marco Ajmone, Pablo Rojo, Manuel Cebrián, Andrés Ortega, Emilio Amaya, Mattis Choummanivong, Alexander Lynch, Andrés Carrillo and Simon Bihoreau. The `fastautoml` library described in Chapter 16 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.

Finally, I would like to give thanks to my parents who gave me the opportunities in life they didn't have, and to my wife and my three kids who provide ultimate meaning to my life.

## **Disclaimer**

I would like to leave it clear that the overwhelming majority of the ideas contained in this book are not mine. The intuition comes from the great minds of our history: Occam, Llull, Leibniz, Newton, etc., and from philosophers like Plato, Popper, Feyerabend, Wittgenstein, etc. Also, I have leveraged on the mathematical theories developed by scientists like Turing, Church, Post, Shannon, Solomonoff, Chaitin, Kolmogorov, and many others. My only original contribution with this book is that I have, perhaps, connected some dots, and that I have provided a, perhaps, interesting reinterpretation of some old ideas. In the References sections included at the end of each chapter there is a description of the works I have used to write the book. In the text I use passive voice ("it is defined") whenever I know that a concept is not mine, and active voice ("we define") when I am not aware of a previous use of this concept by somebody else.





## 1. Introduction

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

Occam's razor principle

The quest for knowledge usually starts by identifying a set of entities we would like to understand. The elements of this set can be almost anything. If we are mathematicians, our set of interest will be composed by mathematical concepts; if we are biologists, the set will be living things; and if we are engineers, it will be machines that solve problems. Our goal, as researchers, is to understand as much as possible about those entities. We want to understand how things work because that allows us to forecast the consequences of our actions. For example, we know that if we apply a sufficient amount of heat to a pile of wood, a fire will start. Also, and more challenging, looking at consequences we can try to infer the causes; if I have fever, maybe I have been infected by a virus. Understanding is how we solve problems in practice, and understanding means to find patterns or regularities that allows us to build a simplified description, or model, of the original entities so we can manipulate them to our convenience.

Ideally, we would like that given our descriptions we should be able to reconstruct the original entities under study. However, for the majority of entities this is not possible, like for example with the abstract ones. Instead, what we have to do is to work with representations, that is, use texts or data that try to capture as many details of the original entities as possible. If we are physicists the representation of an entity could be the result of an experiment, if we are computer scientists it could be a dataset composed by measurements, or if we are sociologists what we might have is a collection of observed facts. In Figure 1.1 it is depicted this process: we would like our descriptions to model entities, but what they model is our artificial representation of those entities.

The question is how well our descriptions explain the original entities through the modeling of the representations that encode those entities. There are multiple sources of error we should consider. The first one is that our encoding method might not be perfect. That is, if the ideal representation of an entity  $e$  is the string  $r$ , in practice we could be working with another string  $r'$  that is close to  $r$ , but not equal. We call this type of error *messaging*. The second source of error

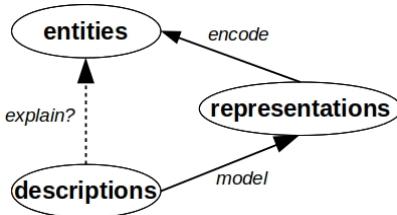


Figure 1.1: The Problem of Understanding.

is due to the fact that our descriptions are not perfect. Ideally a description  $d$  should allow us to recreate the string  $r'$  (the string  $r$  is normally unknown), but in practice it produces another string  $r''$  which is, again, close to  $r'$  but not equal. This second type of error is what we call *inaccuracy*. Finally, the third group of errors deals with the descriptions themselves. Given the limited cognitive capabilities of humans, we are interested in the shortest possible description  $d$  of  $e$ , so we can understand the entity, make predictions, or derive consequences. However, our current best known description  $d'$  probably will be longer than  $d$ . This third type of error is what we call *surfeit*. Finally, we combine these three types of errors, miscoding, inaccuracy and surfeit, in a single quantity called *nescience*, as a quantitative measure of how much we do not know about an entity.

This chapter contains a gentle introduction to our new *theory of nescience*, and a short review of the main results of this theory. The chapter avoids the use of advanced mathematics, but concepts are still semi-formally described. Also, in order to clarify the new ideas introduced, we provide multiple practical examples.

## 1.1 Entities

Our theory starts by assuming there exists a non-empty collection of *entities*, denoted by  $\mathcal{E}$ , that are the subject of past, present or future scientific research. And, of course, we assume that (at least some of) these entities can be fully understood and described by science. Technically speaking  $\mathcal{E}$  is not a well defined set, since the only restriction is that it must be nonempty. The abstract nature of  $\mathcal{E}$  has some advantages, but also introduces important limitations. The main limitation is that our definition of nescience is, in general, a non-computable quantity, and so, it must be approximated in practice. The main advantage is that we can apply the new concepts and methods introduced in this book to other domains, not only to the discovery of solutions to open problems.

In the theory of nescience we do not allow universal sets, that is, we cannot assume the existence of a set  $\mathcal{E}$  that contains everything. The problem of universal sets is that they violate Cantor's theorem (see Section 9.1). Cantor's theorem states that the set  $\mathcal{P}(\mathcal{E})$  composed by all the possible subsets of  $\mathcal{E}$  has more elements than the original set  $\mathcal{E}$ , and this is a contradiction with the fact that  $\mathcal{E}$  contains everything. In the theory of nescience  $\mathcal{E}$  must be the set of "something". Moreover, not all possible sets are allowed. For example, the Russel's paradox propose to consider the set  $\mathcal{E}$  composed by all those sets that are not members of themselves; the paradox arises when we try to answer the question of whether  $\mathcal{E}$  is a member of itself or not (see Section 9.1). We require from every set  $\mathcal{E}$  to be restricted to one particular type of elements, so they cannot be members of themselves.

Usually the set  $\mathcal{E}$  corresponds to an individual domain of knowledge, and its constituent elements depend on how the theory is being applied in practice. Examples of such sets of entities could be: a collection of mathematical objects (abstract); the kingdom of animalia (living things); known and unknown human needs (abstract); all possible computer programs (strings of symbols), etc.

## 1.2 Representations

Most of the entities can not be the subject of a direct scientific analysis because, for example, they are abstract objects. Instead, what we have to do is to work with representations of those entities. Let  $\mathcal{R}_{\mathcal{E}}$  be the collection of finite binary strings, called *representations*, that encode the entities of  $\mathcal{E}$ . The exact format of these strings is something that depends on the entities in which the theory of nescience is being applied. In some cases, entities will be strings themselves (e.g. computer programs), and in others they will be abstract objects that have to be encoded as strings (e.g. human needs). Note that an entity  $e \in \mathcal{E}$  might have more than one valid representation in  $\mathcal{R}_{\mathcal{E}}$ . How to encode abstract entities as strings of symbols in such a way that they capture all the details and nuances of the original entities is a difficult, still unsolved, problem, and so, the set  $\mathcal{R}_{\mathcal{E}}$  is normally unknown.

Ideally, we would like to have an encoding function  $f : \mathcal{E} \rightarrow \mathcal{R}_{\mathcal{E}}$  from a set of entities  $\mathcal{E}$  to the set of all possible representations. Unfortunately, in practice, it is not possible to define such a function, since as we have already mentioned, the set  $\mathcal{E}$  is not a well defined set, that is, we cannot tell what it is and what is not a member of this set. For example, it requires a lot more research before we can tell exactly what a human need is.

From a theoretical point of view we could take advantage of the concept of *oracle Turing machine* (see Chapter 4) to address this problem. A Turing machine is a mathematical model of a computer, and an oracle Turing machine is somehow like a mathematical model of a computer connected to Internet. We could assume that this computer can query an hypothetical external service, hosted somewhere in another computer, to check if a given string  $r$  encodes *any* entity of  $\mathcal{E}$ . We cannot ask the oracle if the string  $r$  is a valid representation of the entity  $e$  in which we are interested, because that would require to provide a valid representation of  $e$  as a string of symbols, something that we cannot do since, in general, it is unknown. The concept of oracle machine allow 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 point of view, we usually approximate the set  $\mathcal{R}_{\mathcal{E}}$  by another set  $\hat{\mathcal{R}}_{\mathcal{E}} \subseteq \mathcal{B}^*$  of strings that we believe are good enough representations of the entities of  $\mathcal{E}$ . In science, traditionally, these representations have had the form of images or drawings (e.g. biology), collection of facts (e.g. sociology), or the result of experiments (e.g. physics). Recently, and due to the huge advances in the capacity of computers to collect and store data, a new and powerful way to encode entities has emerged: the use of large collections of data as representations. Please mind that in the process of encoding we are not interested in finding the shortest possible representation of the entities, what we need is high quality representations.

We should be aware that in many practical problems, the selected representations of abstract entities will not be able to fully capture all the details of the original objects. That is, we are dealing with simplified abstractions of reality that might limit our capability of making universal statements about nature (see Chapter 10).

■ **Example 1.1** When the topics under study are animals (the set  $\mathcal{E}$ ), we could use as representations a binary encoding of their DNAs (the set  $\mathcal{R}_{\mathcal{E}}$ ). Although today we do not have the technology to grow a creature given its DNA, in theory it could be possible to do so. However, the DNA is not enough to fully reconstruct the original animal, since we also need to know what happened along its life. For example, what if we are dealing with a cat with only three legs because it lost one in an accident? That information is not included in its DNA. If we are interested in studying the characteristics of some species, it will be sufficient to work with the DNA of some representative sample of individuals that belongs to each species. However, if we are studying the particular individuals inside a species, we also need a way to encode the history of each animal, or a way to encode the details not covered by the DNA. ■

A consequence of working with strings as representations (the set  $\mathcal{R}_{\mathcal{E}}$ ) is that it might happen

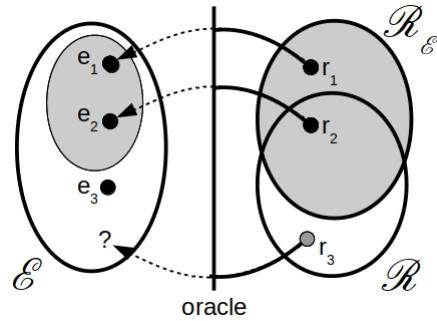


Figure 1.2: Entities and Representations.

that there exist entities that are not encoded by any representation (see the gray areas in Figure 1.2, in particular, the entity  $e_3$  is not encoded by any representation). For example, if our set of entities is the set of real numbers, it turns out that there exist numbers that do not have a finite representation (we do not allow infinite strings as representations). Intuitively, we could say that for many domains of knowledge the number of problems is greater than the number of solutions. A consequence of working with approximations of representations (the set  $\mathcal{R}$ ) is that some representations might encode the wrong entities (the representation  $r_3$  in Figure 1.2). This is due to the fact that our knowledge is incomplete, and we might be using incorrect representations for the entities of  $\mathcal{E}$ . Working with incomplete knowledge has an additional problem, there may be some entities whose existence we do not know yet, what we call the *unknown unknown*. For example, the representation  $r_1$  in Figure 1.2 is not part of the set  $\mathcal{R}$ , and so, it is not currently considered by researchers even though it represents a valid entity  $e_1$ . We are interested in investigating a procedure to discover new, previously unknown, research entities from the set  $\mathcal{R}_{\mathcal{E}}$  (see Section 1.11 and Chapter 14).

A *research area*  $A$  is subset of topics, that is  $A \subset \mathcal{R}_{\mathcal{E}}$ . In practice, areas are useful as long as the topics included in the area share a common property. We might be interested in studying what we do not know about a particular area. For example, if the set of topics is "animals", examples of areas could be "invertebrates", "mammals", "birds", "amphibians", "reptiles" and "fish". In order to measure what we do not know about an area, first we have to provide a description of the topics included in that area. However, in general, our knowledge about the topics that compose an area is limited, and so, we can only partially describe them. Moreover, as our understanding of a research area changes, the number of topics included in its known subset changes as well. Consequently, the properties of areas can be studied only relative to our current knowledge.

■ **Example 1.2** If our set of topics is "astronomy", a research area could be the set of "habitable planets", from which we currently known only a few of them. A model of this area would be a description of those already known habitable planets. ■

Fix figure and the claim that there are strings that do not represent any entity.

### 1.3 Descriptions

Once we have identified the set  $\mathcal{R}$  of possible representations, we have to provide a way to describe them, that is, to formulate our theories or models about how things work. As humans, we have very limited cognitive capabilities, and so, we rely in the use of simple models of nature in order to understand observed facts, and to forecast the results of our own actions.

What constitutes a valid description of an entity is a difficult, still unsolved, problem. For example, the Berry paradox proposes the following description "the smallest positive integer not describable in fewer than twelve words". The paradox arises because we have just described

this number with only eleven words. In order to avoid these kind of paradoxes, in the theory of nescience we require that a description for an entity must be a finite string of symbols from which we can fully and effectively reconstruct one of the possible representations of the original entity. By "effectively reconstruct" we mean that our models must be computer programs that when executed they print the selected representation. Since Newton, science is about mathematical models, that is, the search of a set of functions that fully describe the objects found in Nature and their relations. In the theory of nescience we go one step beyond and require that those models must be computable.

Descriptions are composed by two parts, a Turing machine  $TM$  (a computer program) that implements all the regularities found in the entity's representation (the compressible part), and an input string  $a$  to this program that contains a literal description of what is left (the non-compressible part). From a formal point of view, a description for a representation  $r \in \mathcal{R}$  is a string  $\langle TM, a \rangle$  such that  $TM(a) = r$ . This two part nature of descriptions somehow resembles our classical distinction between theories and assumptions, theories and initial conditions, problems and particular instances of problems, species and individuals, etc. For example, a description could be composed by a set of differential equations modeling a system (the compressible part), and the collection of initial conditions (the non-compressible part). The actual interpretation of the pair  $\langle TM, a \rangle$  is something that depends on the particular details of the set of entities in which that theory is being applied.

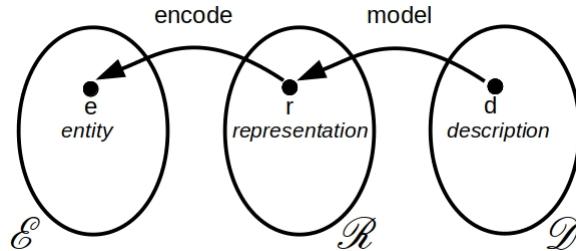


Figure 1.3: Entities, representations and descriptions.

In Figure 1.3 it is shown graphically the relation between entities, representations and descriptions (the set of all possible descriptions is denoted by  $\mathcal{D}$ ). Not every possible string is a description, since we require that a descriptions must be based on Turing machines, and not all possible descriptions describe valid representations.

**■ Example 1.3** From an intuitive point of view, if we are studying how our world works, from a physical macroscopic point of view, our set of possible descriptions would include, among others, the following elements: Aristotelian physics, Cartesian physics, Newtonian physics, Einstein's theory of general relativity and superstring theory. Our current best description would be Einstein's theory of general relativity (superstring theory has not been validated experimentally yet). ■

A representation  $r \in \mathcal{R}$  can have multiple descriptions, and so, the goal of science is to find the shortest possible description, denoted by  $d^*$ , that allows us to fully reconstruct the representation  $r$ . Unfortunately, there does not exist a method or algorithm that given a string, returns the shortest possible computer program that prints that string (see the result of incomputability of Kolmogorov complexity in Chapter 6). In particular, there does not exist a method that given a representation finds the shortest possible description. That is, science is a non-computable problem and so, we have to find a collection of heuristics that allow us to approximate the optimal solution. This collection of heuristics is what we call a *scientific method*.

In the theory of nescience we are interested in understanding, and quantitatively measuring, what can be wrong in the ideal process depicted in Figure 1.3. In Sections 1.4, 1.5, and 1.6 we will propose a collection of metrics to measure each possible source of error, and in Section 1.7 we will see how to combine all these concepts into a single quantity, called *nescience*. The new metric of

nescience allow us to quantitatively measure how much we do not know about a research entity.

## 1.4 Miscoding

As we have seen, for the majority of the scientific disciplines, the set  $\mathcal{E}$  of entities under consideration could be composed by abstract elements, or other kind of objects that are not easily represented as a string of symbols. Also, it might happen that our current understanding of the elements of  $\mathcal{E}$  is limited, and so, we can not properly encode them. In those cases we have to work with approximate representations instead of the valid representations. We are interested in knowing the error introduced due to the use of these incorrect representations.

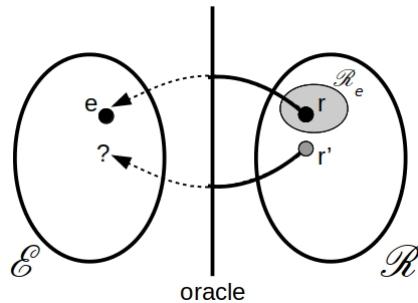


Figure 1.4: Miscoding of topics.

We propose to quantify the *miscoding* of an incorrect representation  $r'$  as the length of the shortest computer program that can print the correct representation  $r$  having as input  $r'$ , denoted by  $K(r|r')$ . See Figure 1.4, where  $\mathcal{R}_e$  denotes the set of strings that correctly represent the entity  $e$ . Intuitively, miscoding measures the effort (measured as the length of a program, not as the time it takes to that program to do its work) required to fix the incorrect representation. For example, if our representation contains some errors, the program should be able to identify and correct those errors; or if our representation is missing some critical information required to fully encode the entity, the program will contain, hard-wired, that missing information.

However this approach is not sufficient to fully capture our intuitive notion of miscoding. The problem arises when our representation  $r'$  contains additional information that is not needed to encode the entity  $e$ . In this case, it might happen that our descriptions include elements modeling those unnecessary symbols, making them artificially long. For example, in a experiment we could be measuring features that do not influence at all the result of the experiment, and given our limited understanding of the original entity, our current description could include those features as predictors<sup>1</sup>. In order to avoid this problem, we have to compute also the length of the shortest computer program that can print the incorrect description  $r'$  having as input the correct one  $r$ , that is,  $K(r'|r)$ . Then, miscoding could be defined as the maximum of these two lengths  $\max\{K(r|r'), K(r'|r)\}$ .

Unfortunately, this last definition still presents some practical problems. Most of the entities have multiple valid representations. Maybe  $r'$  is far from a representation  $r_1$ , but it is closer to a second one  $r_2$ . It would be unfair to say that a description  $d$  that perfectly model  $r_2$  is a bad descriptions because it does not model  $r_1$ , since both,  $r_1$  and  $r_2$  represent the same entity  $e$ . A possible solution to this problem would be to look at  $\min_{r \in \mathcal{R}_e} \{\max\{K(r|r'), K(r'|r)\}\}$ .

■ **Example 1.4** Let  $e$  be the abstract entity called "Pi constant", that is, the ratio of a circle's circumference to its diameter; let  $r$  be the Wallis' product  $2(\frac{2}{1} \cdot \frac{2}{3} \cdot \frac{4}{3} \cdot \frac{4}{5} \cdot \frac{6}{5} \dots)$ , one of the valid

<sup>1</sup>TODO: This example is confusing, since the experiment describes a cause-effect situation, and our machine learning algorithm could identify non-relevant features automatically. Use a different example.

representations of  $e$ ; and let  $d$  be the description  $\sum_{n=0}^{\infty} \frac{(-1)^n}{2n+1}$ . It would be unfair to affirm that  $d$  is a terribly bad description for the entity  $e$  because it does not print  $r$ . In fact,  $d$  produces the Leibniz's series  $4(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots)$  that it is also a valid representation of Pi. Leibniz's series should not be classified as a miscoded representation of Pi, even in the hypothetical case that it were unknown by mathematicians. ■

As example 1.4 pointed it out, one of the most difficult problems with the definition of miscoding is that the set  $\mathcal{R}_e$  of valid representations for the entity  $e$  is, in general, not known. From a theoretical point of view we could resort again to the oracle Turing machine to address this problem. However, as we have seen, we cannot ask the oracle if the string  $r$  is a valid representation of the entity  $e$  in which we are interested (the set  $\mathcal{R}_e$ ), since that would require to provide a valid representation of  $e$  as a string of symbols, something that, in general, cannot 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  $\mathcal{R}_{\mathcal{E}}$ .

Taking into account all the above considerations, we define the miscoding of a representation  $r$ , denoted by  $\mu(r)$ , as:

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

where  $\min^o$  means that the minimum function has to be computed by an oracle. Note that in the definition we have introduced the normalization factor  $K(r_e)$ , that is, the length of the shortest program that can print  $r_e$ , because we are interested in comparing the miscoding of multiple, possibly unrelated, representations.

Since miscoding does not take as argument the original entity  $e$ , it may be the case that what we are representing is not what we were expecting. That is, with our representations and descriptions we might be studying a different entity. This is something that happens quite often in the practice of scientific research.

■ **Example 1.5** At the end of the eighteenth century, the chemist Joseph Priestley thought he was studying the nonexistent entity called "phlogiston", a fire-like element that was supposed to be contained within combustible bodies and released during combustion. It turned out that he was studying a completely different entity called "oxygen". ■

According to the theory of nescience, our task as researchers is not only to find the proper representations of the entities of  $\mathcal{E}$ , but also to discover how this ideal oracle machine that knows how to properly encode entities works. That is, why our representations constitute good encodings of the entities under study.

## 1.5 Inaccuracy

In the previous section we have characterized how much we do not know about an entity  $e$  in terms of the miscoding due to using a wrong representation  $r'$  instead of a correct one  $r$ . In this section we are interested in how much we do not know given a description. Ideally we should be working with a description  $d$  that allows us to fully reconstruct the representation  $r'$  (recall that the representation  $r$  might be unknown), but in general this is not the case in practice. Normally we will be working with a description  $d'$  that prints out a string  $r''$  (remember that we require that descriptions must be computer programs) that it is (hopefully) close to the string  $r'$ , but not equal. In this case we say that the description  $d'$  is an *inaccurate* description of the representation  $r'$  (see Figure 1.5).

If a description is inaccurate for a representation, we would like to have a quantitative measure of how far we are from properly modeling the representation. A natural way to define this measure would be by means of computing the effort required to fix the output of our inaccurate description.



Figure 1.5: Inaccuracy of a description.

In this sense, inaccuracy could be given by the length of the shortest computer program that can print the representation given as input the wrong representation produced by the description. However, as it was the case of miscoding, in order to have a complete picture of the error made with the description  $d$  we have to compute also how difficult is to print the inaccurate representation given the good one. It might happen that our description  $d$  is modeling things that have nothing to do with the representation  $r'$ , and simply ignoring those elements is not a solution.

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  $r'$ , denoted by  $\iota(d, r')$ , as:

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

Again, since what we need is the broadest possible interpretation of the concept of inaccuracy, we have to divide by the normalization factor  $\max\{K(r'), K(r'')\}$ , so we can compare multiple descriptions for the same representation as well as descriptions for different representations.

We prefer the term *inaccuracy* over the term *error*. Error is given by two components, precision and accuracy. But precision mostly refers to continuous systems, and since we are dealing with discrete strings of symbols, it does not make too much sense to talk about precision in this context.

In practice it is a difficult task to compute the inaccuracy associated with the description of a representation, since, as we have already said, finding the length of the shortest computer program that can print a string is a non-computable problem. If the original entities are texts themselves, the inaccuracy could be approximated using compression algorithms, where the Kolmogorov complexity is approximated with the length of the compressed text using a compressor. If topics are abstract entities, like mathematical concepts, their descriptions could be based on the result of an experiment, and so, the inaccuracy could be based on the error of the model (for example, by means of computing the length of the additional information required to fully describe the results of the experiment given the model). In this sense, our definition of inaccuracy is a generalization of the concept of error, since it can be applied to multiple types of entities, not only to those entities that can be encoded as datasets.

■ **Example 1.6 TODO: Review this example.** The topic Newton's second law of motion  $F = ma$  could be encoded given the results of an experiment using objects of different masses to which we apply different forces and then we measure the acceleration. The dataset would be composed by the masses, the forces, and the accelerations achieved for each combination of mass and force. However, if we are interested in the acceleration due to gravity, forces and masses cancel out, and so, the only value to measure is acceleration. Of course, if the result of the experiment is a large collection of measurements, the encoding of the entity would be very long. But, as we said before, with the encodings of entities we are not interested in finding the shortest possible strings, what we are looking for are complete and accurate encodings of topics. For this example we will 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 is composed by 81 measurements, and each value is

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. In order to encode the dataset we require 20 bits per measure (using an uniform code, as it is explained in Chapter 5), and so, to encode the full dataset we require 1,620 bits. If we assume that our model predicts a gravity of  $980,000 \text{ cm/s}^2$  plus a random error that follows a normal distribution (estimated using a maximum likelihood approach), we have that it requires 453 bits to encode the data given the model (see Chapter 16 for more information about how to encode a dataset given a model). We have that the approximated inaccuracy of the model 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 (miscoding).

## 1.6 Surfeit

If it requires a lot of time and effort to explain how something works, we probably do not understand it, and our knowledge must be incomplete. Long descriptions usually contain a lot of things that are not needed, and, in our opinion, one of the main tasks of science should be to reduce as much as possible those unnecessary elements. We relay on descriptions, usually mathematical models, to forecast the future given the past, to understand the relation between causes and effects, and to design machines that solve problems. We have a strong interest in finding the shortest possible models that describe how things work, so they can fit in our limited brain and make our work as scientists and engineers easier<sup>2</sup>.

The theoretical limit of what can be known about a representation, that is, its perfect description  $d^*$ , is given by the shortest possible computer program that allows us to reconstruct the representation. The surfeit of a description  $d'$  can be computed by comparing the length of this particular description with the length of the best possible description  $d^*$  for that representation (see Figure 1.6).

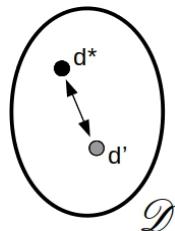


Figure 1.6: Surfeit of a model.

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

<sup>2</sup>In a (hopefully) not so long distant future, when all scientific reasoning is performed by computers, having the shortest possible models will no be a priority any more, and it will become a mere scientific curiosity of limited practical value.

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.

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 6.5), 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.

Randomness effectively imposes a limit to how much we can know about a particular entity. Far from being a handicap, the proper understanding of this absolute epistemological limit is an opportunity to advance in our understanding of open problems. Moreover, since what is limited is the enhancement of our knowledge about already identified research entities, as opposed to the discovery of new ones (new entities are understood to be infinite), we can also apply our understanding of randomness to discover new research entities.

With our definition of surfeit, where longer explanations are worse, we are not suggesting that textbooks should always be as concise as possible. On the contrary, in some situations we expect textbooks to be highly redundant. A concise book is a text that contains a large amount of information in a very reduced space, and so, it is very difficult for humans to assimilate (understand) that information. However, a redundant textbook (like this one) contains the same amount of information but in a larger space, and so, it is easier to grasp its contents. Moreover, in some knowledge areas other than science, redundancy could be desirable. For example, in law, redundancy helps lawyers memorize legal texts, and in music, repetition could be a good thing for harmony, for example in a canon.

The concept of surfeit can be extended to research areas as well, in such a way that we can not only compute the surfeit of a given area as a whole, but also compare the surfeit of multiple, unrelated, areas.

■ **Example 1.7** Based in the concept of "category" of Wikipedia, that it is quite similar to our concept of "research area", we could compute the surfeit of the major scientific areas (see Chapter 12 for more information about how to compute surfeit in practice). For example, as it is shown in Table 1.1<sup>3</sup>, in general our understanding of Mathematics is higher than our understanding of Computer Science, since the redundancy of Mathematics is 0.351 and the redundancy of Computer Science is 0.443. The concept of redundancy applied to areas largely matches 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

---

<sup>3</sup>Data from October 2014.

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

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). 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 16). 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

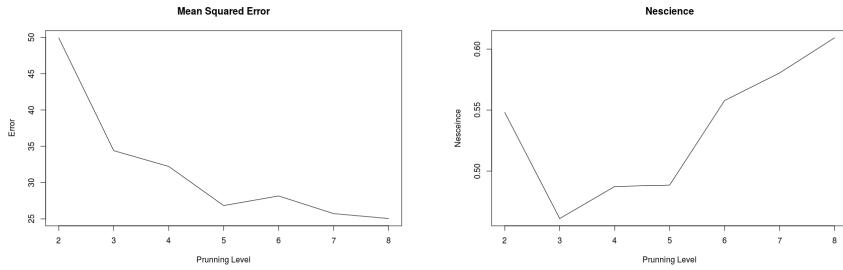


Table 1.2: RMSE and Nescience of Models

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 the topic does not increase<sup>4</sup>. ■

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 encyclopedia<sup>5,6</sup>. 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 18 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. ■

<sup>4</sup>Please 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.

<sup>5</sup>[https://en.wikipedia.org/wiki/Causes\\_of\\_World\\_War\\_I](https://en.wikipedia.org/wiki/Causes_of_World_War_I) (retrieved on August 2017)

<sup>6</sup>[https://en.wikipedia.org/wiki/Lotk-Volterra\\_equations](https://en.wikipedia.org/wiki/Lotk-Volterra_equations) (retrieved on August 2017)

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 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 SQLite<sup>7</sup>. 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 17 we will see that this is not the case for most of the existing software platforms and applications. ■

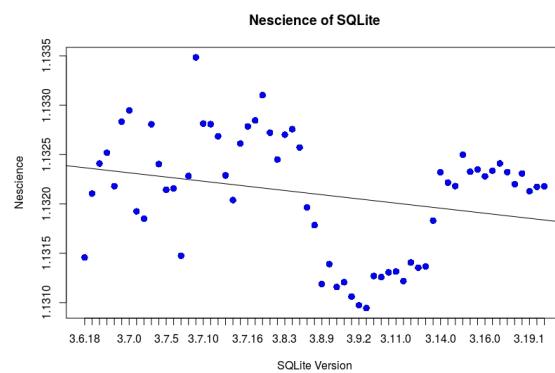


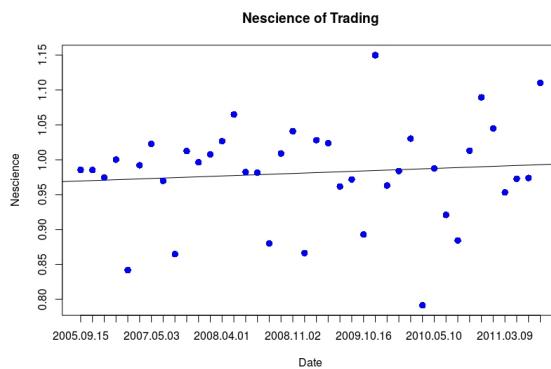
Figure 1.7: Evolution of Nescience

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

<sup>7</sup>[www.sqlite.org](http://www.sqlite.org)

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 17.3 for 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. ■



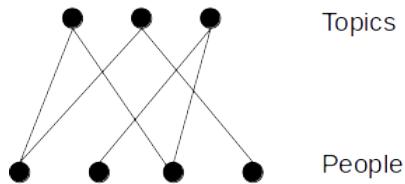


Figure 1.9: Relevance Graph

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.

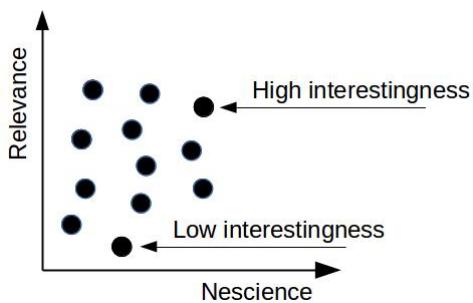


Figure 1.10: Interestingness of Topics

■ **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 organized<sup>8</sup> 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

<sup>8</sup>Data from November 2014.

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 18 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 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.

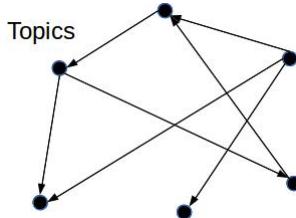


Figure 1.11: Applicability Graph

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 disease<sup>9</sup>. 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.

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).

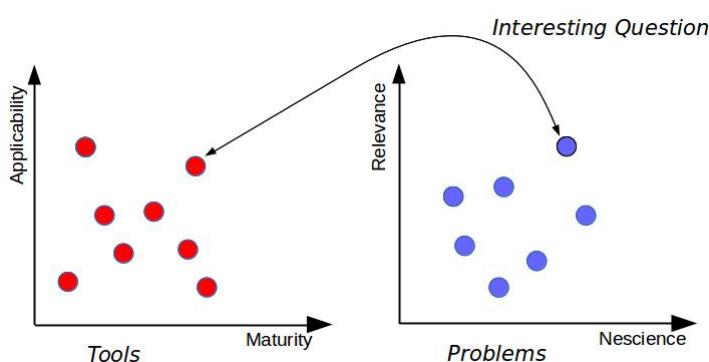


Figure 1.12: Interesting Questions

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  $\langle TM, a \rangle$ ,

<sup>9</sup>We 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.

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 14), 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)<sup>10</sup>. 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.

<sup>10</sup>As we will see in Chapter 9, we require  $\mathcal{R}$  to be closed under the operation of concatenation of representations, that

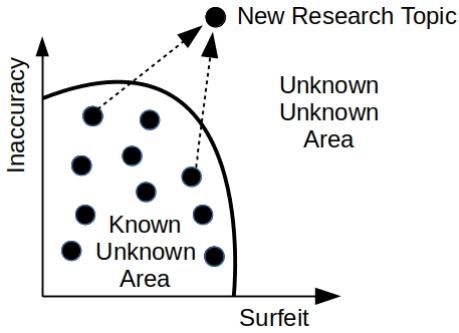


Figure 1.14: New Research Topics

■ **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 [Mos16]). The gravity dataset used in Example 1.6 comes from [Cre82], and a more detailed analysis can be found at [DH97]. The Boston housing dataset referred in Example 1.8 was published in [HR78] and further discussed in [BKW05]; for more information about decision trees see for example [Jam+13]. For more information about trading systems and technical indicators see for example [Kau13], and how to quantitatively test if a system is profitable to [Par92]. The Locka-Voterra model for population dynamics was originally described in [Lot20]. How to apply graph analysis to recommendation engines is covered in [Cor+15]. And finally, if the reader is interested in knowing how far a dog flea can jump, please refer to [CJF00].

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 [Nag14].

---

is, for any two representations  $r, s \in \mathcal{R}$  we have that  $rs \in \mathcal{R}$ .



# Part 1: Background

<b>2</b>	<b>Discrete Mathematics</b>	41
2.1	Sets, Relations and Functions	
2.2	Strings	
2.3	Matrices	
2.4	Graphs	
2.5	Counting Methods	
<b>3</b>	<b>Discrete Probability</b>	49
3.1	Foundations	
3.2	Conditional Probability	
3.3	Random Variables	
3.4	Characterizing Distributions	
3.5	Common Distributions	
3.6	Large Random Samples	
<b>4</b>	<b>Computability</b>	67
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	
<b>5</b>	<b>Coding</b>	77
5.1	Coding	
5.2	Kraft Inequality	
5.3	Entropy	
5.4	Optimal Codes	
5.5	Huffman Algorithm	
5.6	Discretization Algorithms	
<b>6</b>	<b>Complexity</b>	91
6.1	Strings Complexity	
6.2	Properties of Complexity	
6.3	Conditional Kolmogorov complexity	
6.4	Information Distance	
6.5	Incompressibility and Randomness	
<b>7</b>	<b>Learning</b>	99
7.1	Statistical Inference	
7.2	Machine Learning	
7.3	Minimum Message Length	
7.4	Minimum Description Length	
7.5	Multiobjective Optimization	
<b>8</b>	<b>Philosophy of Science</b>	117
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

Most of the mathematics used throughout this book belong to the area of *discrete mathematics*. Discrete mathematics is characterized because it studies mathematical objects that have distinct or separated values, rather than continuous. Examples of discrete objects used in this book are integers, strings, graphs and computer programs. A key distinctive element of discrete sets is that they can be enumerated with natural numbers, that is, they are countable. We will barely use continuous mathematics, for example calculus, in the theoretical developments of the theory of nescience.

Our main interest in discrete mathematics is because computers. The theory of nescience borrows concepts and ideas from multiple areas of computer science: algorithms, coding, string complexity and others. Computers operate in discrete steps and the data processed is stored in discrete units of memory. We are interested in computers because we want to apply our theoretical developments to as many real objects as possible, and we think that the right way to model our world is by means of using computers. In pure mathematics it is customary to discuss abstract objects without worrying about what they represent. In the theory of nescience, on the contrary, the representation (encoding) of objects plays a crucial role.

This chapter is intended as a quick review of the basic concepts of discrete mathematics; no formal definitions are provided and theorems are not proved. Discrete mathematics is an extremely diverse and large area of study. We will review only those elements that are required to understand the theory of nescience. Some of the theories involved (computability, information, complexity, ...) require a deeper coverage, and so, they are studied in separate chapters. In the References section there is a list of suggested books that explain in detail the topics covered in this chapter.

## 2.1 Sets, Relations and Functions

We denote by  $\mathbb{N}$ ,  $\mathbb{Z}$ ,  $\mathbb{Q}$  and  $\mathbb{R}$  the set of *natural numbers* (including 0), *integers*, *rational numbers* and *real numbers* respectively. *Positive integers* are represented by  $\mathbb{Z}^+$ , and *positive reals* by  $\mathbb{R}^+$  (in both cases the number 0 is included). Let  $A$  be a *set*, by  $x \in A$  we mean that  $x$  is a *member* of  $A$ . The members of a set could be listed using brackets, for example  $A = \{0, 1, 2, 3\}$ , or the *set formation* notation  $A = \{x \in \mathbb{N} : x < 4\}$  (when we require the *universe* of the set to be known).

Let  $A$  and  $B$  be two sets,  $A = B$  means that both sets are *equal*,  $A \subset B$  that  $A$  is a *subset* of  $B$  but not equal, and  $A \subseteq B$  that  $A$  is a *subset or equal* to  $B$ . Note that  $A = B$  if, and only if,  $A \subseteq B$  and  $B \subseteq A$ . The *empty set* is denoted by  $\emptyset$ .

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

The *cardinality* of a finite set  $A$ , denoted by  $d(A)$ , is the number of elements of  $A$ . The cardinality of  $\emptyset$  is 0. Given the sets  $A$  and  $B$ ,  $A \cup B$  means the *union* of  $A$  and  $B$ , and  $A \cap B$  the *intersection* of  $A$  and  $B$ . The union of  $n$  sets  $A_1, A_2, \dots, A_n$  is represented by  $\cup_{i=1}^n A_i$ , and their intersection by  $\cap_{i=1}^n A_i$ . In case of an arbitrary collection of sets  $I$  we use  $\cup_{i \in I} A_i$  and  $\cap_{i \in I} A_i$ . And in case of an infinite collection of sets we use  $\cup_i^\infty A_i$  and  $\cap_i^\infty A_i$

Occasionally we will use *Venn diagrams* to represent sets graphically (see Figure 2.1).



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

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  $d(\mathcal{P}(A)) = 2^n$ .

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

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

■

Let  $A$  be a non-empty set and  $\mathcal{F}$  a collection of subsets of  $A$ , the pair  $(A, \mathcal{F})$  is called an *field* over  $A$  if the following properties are satisfied: it contains the empty set  $\emptyset \in \mathcal{F}$ , it is closed under complementation  $F^c \in \mathcal{F}$  for all  $F \in \mathcal{F}$ , and it is closed under finite unions  $F_1 \cup \dots \cup F_n \in \mathcal{F}$  for all  $F_1, \dots, F_n \in \mathcal{F}$ . It can be show that a field must satisfy the following two additional properties:  $A \in \mathcal{F}$  and  $F_1 \cap \dots \cap F_n \in \mathcal{F}$  for all  $F_1, \dots, F_n \in \mathcal{F}$  (closed under finite intersections).

Given  $x$  and  $y$ , the *ordered pair*  $(x, y)$  consists of  $x$  and  $y$  in that order. An *n-tuple* is an *n*-ary ordered pair. The *Cartesian product* of two sets  $A$  and  $B$ , denoted by  $A \times B$ , is the set composed by all the ordered pairs  $(x, y)$  such that  $x \in A$  and  $y \in B$ . The extension of the Cartesian product to  $n$  sets  $A_1, A_2, \dots, A_n$  is represented by  $A_1 \times A_2 \times \dots \times A_n$ , and the *n-fold Cartesian product* of  $A$  with itself is denoted by  $A^n$ .

A subset  $R \subseteq A \times A$  is called a *binary relation*; we use the notation  $aRb$  to denote that  $(a, b) \in R$ . A binary relation is called *reflexive* if for all  $a \in A$  we have that  $aRa$ . A binary relation is called

*symmetric* if for all  $a, b \in A$  we have that if  $aRb$  then  $bRa$ . A binary relation in which  $aRb$  and  $bRa$  implies that  $a = b$  is called *antisymmetric*. A binary relation is *transitive* if for all  $a, b, c \in A$  we have that if  $aRb$  and  $bRc$  then  $aRc$ . A binary relation is called *total* if for all  $a, b \in A$  either  $aRb$  or  $bRa$ . Binary relations can be extended to two different sets  $A$  and  $B$  as a subset  $R \subseteq A \times B$ , and to  $n$ -ary relations  $R \subseteq A_1 \times A_2 \times \dots \times A_n$ .

A binary relation  $R \subseteq A \times A$  that is reflexive, symmetric and transitive is called an *equivalence relation*. Equivalence relations are denoted by  $\sim$ . Two elements  $a, b \in A$  are said to be *equivalent* if they are related  $a \sim b$ . The *equivalence class* of  $a$ , denoted by  $[a]$ , is composed by all the elements of  $A$  that are equivalent to  $a$ , that is  $[a] := \{b \in A : a \sim b\}$ . An equivalence relation partitions the underlying set into the *quotient set*  $A/\sim := \{[a] : a \in A\}$ .

A *partial order* is a binary relation which is reflexive, transitive and antisymmetric; partial orders are represented by the symbol  $\preceq$ . A set with a partial order is called a *partially ordered set* (also called a *poset*). An element  $a \in A$  of a partially ordered set is *minimal* if it does not exist another element  $b \in A$  such that  $b \preceq a$ ; an element  $a \in A$  is *maximal* if it does not exist another element  $b \in A$  such that  $a \preceq b$ . A relation that is reflexive, transitive, antisymmetric and total is called a *total order*; total orders are represented by the symbol  $\leq$ . A set paired with a total order is called a *totally ordered set*. Given a totally ordered set  $A$  we denote  $\max(A)$  the *maximum* element of  $A$ , and  $\min(A)$  its *minimum* element.

■ **Example 2.3** Let  $R \subset \mathbb{N} \times \mathbb{N}$  a relation such that  $(a, b) \in R$  if  $a$  divides  $b$ .  $\mathbb{N}$  with  $R$  is a partially ordered set. The number 11 is a minimal element of  $R$ , since 11 is a prime number. ■

A *function* is a binary relation  $f \subseteq A \times B$  where for every  $x \in A$  there is at most one  $y \in B$  such that  $(x, y) \in f$ , the elements  $(x, y) \in f$  are denoted by  $f(x) = y$ , and the function by  $f : A \rightarrow B$ . The set  $A$  is called the *domain* of  $f$ , and  $B$  the *codomain*. The set  $\{y \in B : \exists x \in A, f(x) = y\}$  is the *range* of  $f$ . A function is called *partial* if the relation is not defined for all  $x \in A$ ; we denote by  $f(x) \uparrow$  if the function  $f$  is not defined for  $x$ .

■ **Example 2.4** In Section 4.4 we will see an alternative definition of the concept of function, as a procedure, or collection of steps, that assigns to the elements of  $A$  an element of  $B$ . For example, the following C code defines a partial function from  $\mathbb{R}$  to  $\mathbb{R}$ . It is partial because  $inv(0) \uparrow$ :

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

■

A function is *injective* if for all  $x$  and  $y$  if  $f(x) = f(y)$  then  $x = y$ . A function is *surjective* if for all  $y$  there exists at least one  $x$  such that  $f(x) = y$ . A function is *bijective* if it is both, injective and surjective. The *identity* function  $I_A : A \rightarrow A$ , defined by  $f(a) = a$  for all  $a \in A$ , is bijective. The concepts of function, partial function, injective, surjective and bijective can be easily generalized to  $n$ -ary relations.

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

An infinite set  $A$  is *countable* if there exists a bijective function between the elements of  $A$  and the set of natural numbers  $\mathbb{N}$ . A set is *uncountable* if it is neither finite nor countable. We say that a set has *countable many* elements if it is either finite or countable.

■ **Example 2.5**  $\mathbb{N}$ ,  $\mathbb{Z}$  and  $\mathbb{Q}$  are countable sets,  $\mathbb{R}$  is not. ■

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

Given a real number  $x \in \mathbb{R}$ , the *absolute value* of  $x$ , denoted  $|x|$ , is defined as  $x$  if  $x \geq 0$  and  $-x$  if  $x < 0$ . The *ceil* of  $x$ , denoted  $\lceil x \rceil$ , is the least integer that is greater than or equal to  $x$ ; the *floor* of

$x$ , denoted by  $\lfloor x \rfloor$ , is the greatest integer that is less than or equal to  $x$ . Given two positive integers  $a$  and  $b$ ,  $a$  modulo  $b$ , denoted by  $a \bmod b$ , is the remainder of the integer division of  $a$  by  $b$ .

Let  $f$  and  $g$  be two functions  $f, g : \mathbb{N} \rightarrow \mathbb{R}^+$ . We say that  $f(n)$  is of order of  $g(n)$ , denoted by  $f(n) = O(g(n))$ , if there exists positive integers  $c$  and  $m$  such that for every integer  $n \geq m$  we have that  $f(n) \leq cg(n)$ . When  $f(n) = O(g(n))$  we say that  $g$  is an upper bound for  $f$ .

## 2.2 Strings

Let  $\mathcal{S} = \{s_1, s_2, \dots, s_q\}$  be a non-empty finite set called *alphabet*. A *sequence* over  $\mathcal{S}$  is any ordered collection of symbols  $x_1x_2\dots x_n$  from  $\mathcal{S}$ . When the alphabet is the set  $\mathcal{B} = \{0, 1\}$ , the sequences are called *binary*. If the sequence is finite we call it *string*. In this book we will be working most of the time with binary strings. The *length* of a string  $s$ , denoted by  $l(s)$ , is the number of symbols in  $s$ . The *empty string* is the unique string over  $\mathcal{S}$  of length 0, and is denoted  $\lambda$ . Let  $x \in \mathcal{S}$ , by  $x^n$  we denote the string  $xx\dots x$  ( $n$ -times). If  $s = x_1x_2\dots x_n$  is a string, its *reverse*  $s^R$  is  $x_nx_{n-1}\dots x_1$ .

Let  $\mathcal{S}^n$  denote the set of all strings  $s_1s_2\dots s_n$  of length  $n^1$ ,  $\mathcal{S}^+ = \cup_{n \geq 1} \mathcal{S}^n$  and  $\mathcal{S}^* = \mathcal{S}^+ \cup \{\lambda\}$ . Note that all the strings that belong to  $\mathcal{S}^*$  have finite lengths.  $\mathcal{S}^*$  is called the Kleene *closure* of  $\mathcal{S}$ .

■ **Example 2.6** The following relations hold:  $d(\{s \in \mathcal{B}^* : l(s) = n\}) = 2^n$  and  $d(\{s \in \mathcal{B}^* : l(s) \leq n\}) = 2^{n+1} - 1$ . ■

For any two strings  $s$  and  $t$  in  $\mathcal{S}^*$ , the *concatenation* of  $s$  and  $t$ , denoted by  $st$ , is defined as the sequence of symbols in  $s$  followed by the sequence of symbols in  $t$ . Note note that  $l(st) = l(s) + l(t)$ .  $\mathcal{S}^*$  is closed under the operation of concatenation.  $\mathcal{S}^*$  with the operation of concatenation forms a *free monoid*, that it, it is associative  $s(tr) = (st)r$ , and has an identity element  $\lambda a = a\lambda = a$ .

A string  $s$  is said to be a *substring* of  $t$  if there exist (possibly empty) strings  $u$  and  $v$  such that  $t = usv$ . A string  $s$  is said to be a *prefix* of  $t$ , denoted by  $s <_p t$ , if there exists a string  $u$  such that  $t = su$ . A subset  $S \subset \mathcal{S}^*$  is *prefix free* if for all  $s, t \in S$ , if  $s <_p t$  then  $s = t$ . Given a string  $s \in \mathcal{S}^*$ , the *self delimited* version of  $s$ , denoted by  $\bar{s}$ , is  $\bar{s} = 1^{l(s)}0s$ . Note that  $l(\bar{s}) = 2l(s) + 1$ .

■ **Example 2.7** The set  $\bar{\mathcal{S}}^*$  composed by all the self delimited strings of  $\mathcal{S}^*$  is prefix free. ■

If  $\mathcal{S}$  is a total ordered set, we can define a total order on  $\mathcal{S}^*$ , called *shortlex ordering*, in which sequences are primarily sorted by length with the shortest sequences first, and sequences of the same length are sorted into their lexicographical order.

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

Given an arbitrary object  $O$  we denote its representation as a string as  $\langle O \rangle$ , assuming that there exists an standard encoding method. Given the objects  $O_1, O_2, \dots, O_k$ , the concatenation of the representations of these objects  $\langle O_1 \rangle \langle O_2 \rangle \dots \langle O_k \rangle$  is denoted by  $\langle O_1 O_2 \dots O_k \rangle$ . The concatenation of the representation of these objects in such a way that we can decode and uniquely identify all of them is represented by  $\langle O_1, O_2, \dots, O_k \rangle$ , for example, we could use  $\langle O_1, O_2, \dots, O_k \rangle = \langle \bar{O}_1 \rangle \langle \bar{O}_2 \rangle \dots \langle \bar{O}_k \rangle$ .

■ **Example 2.9** Natural numbers can be represented by binary strings using the following encoding method:  $\langle 0 \rangle = \lambda$ ,  $\langle 1 \rangle \rightarrow 0$ ,  $\langle 2 \rangle \rightarrow 1$ ,  $\langle 3 \rangle \rightarrow 00$ ,  $\langle 4 \rangle \rightarrow 01$ ,  $\langle 5 \rangle \rightarrow 10$ ,  $\langle 6 \rangle \rightarrow 11$ ,  $7 \rightarrow 000$ , ... For example  $\langle 3, 7 \rangle$  would be represented as 110001110000. Given this encoding we have that  $l(\langle n \rangle) = \lfloor \log_2(n+1) \rfloor$ . ■

<sup>1</sup>Do not confuse the set of strings of length  $n$  over an alphabet  $\mathcal{S}^n$  with the  $n$ -fold Cartesian product of a set  $\mathcal{S}^n$ , alphabets will be represented by using calligraphic fonts.

### 2.3 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 2.10** 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  $\lambda$  by a matrix  $A$  is the matrix  $\lambda A$  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 sum of 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 2.11** 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  $AB = \begin{pmatrix} 19 & 22 \\ 50 & 58 \end{pmatrix}$ . ■

### 2.4 Graphs

A graph<sup>2</sup>  $G$  is an ordered pair  $(V, E)$  composed by a nonempty set of objects  $V$ , called vertices, connected by set of links  $E$ , called edges. The elements of  $E$  have the form of unordered pairs  $\{u, v\}$  of distinct (no loops allowed) vertices  $u, v \in V$ . Vertices  $u$  and  $v$  are said to be adjacent if there is an edge  $\{u, v\} \in E$ , and if so, they are called the endpoints of the edge. If the set  $V$  is infinite, the graph is called infinite graph; however, in this book we will consider only finite graphs. If the pairs of vertices  $u, v$  are ordered pairs, the graph is called directed graph, and in that case,  $u$

<sup>2</sup>The definition of graph provided in this introduction is equivalent to the definition of simple graph in the standard discrete mathematics literature.

is called the *initial vertex* and  $v$  the *terminal vertex*. If  $G = (V, E)$  is a graph, its *adjacency matrix* is a square  $d(V) \times d(V)$  matrix  $A$  such that the element  $A_{uv}$  is 1 if  $\{u, v\} \in E$  and 0 otherwise.

Graphs are usually depicted as a set of dots for the vertices, joined by lines for the edges. If the graph is directed, we use arrows instead of lines to represent edges.

■ **Example 2.12** Let  $V = \{a, b, c, d\}$  and  $E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{b, c\}\}$ , the graph  $G = (V, E)$  is depicted in Figure 2.2. ■

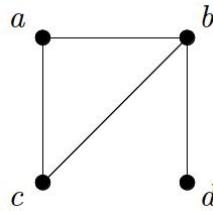


Figure 2.2: An Example of Graph

If  $v$  is an endpoint of an edge  $e$ , then we say that  $e$  is *incident* on  $v$ . The *degree* of a vertex  $v$ , written  $\deg(v)$ , is equal to the number of edges which are incident on  $v$ . A vertex of degree zero is called *isolated*; a vertex of degree one is called *pendant*. The *neighborhood* of vertex  $v$ , denoted by  $N(v)$ , is the set of all vertices that are adjacent to  $v$ . If  $A \subset V$ , the neighborhood of  $A$  is  $N(A) = \cup_{v \in A} N(v)$ . If  $G$  is a directed graph, we call the *in-degree* of a vertex  $v$ , denoted by  $\text{indeg}(v)$ , to the number of edges in which  $v$  is a terminal vertex, and *out-degree*, denoted by  $\text{outdeg}(v)$ , to the number of edges in which  $v$  is an initial vertex. A *path* in a graph is a sequence of distinct vertices  $\{v_0, v_1, \dots, v_k\}$  in which  $v_i$  and  $v_{i+1}$  are adjacent for each  $1 \leq i < k$ . If  $v_0 = v_k$  we say that the path is a *cycle*.

■ **Example 2.13** Given a graph  $G = (V, E)$ , the *handshaking theorem* states that  $\sum_{v \in V} \deg(v) = 2m$ , where  $m = d(E)$ , since each edge has two end points. ■

A graph  $G$  is said to be *bipartite* if the set of vertices  $V$  can be partitioned into two subsets  $V_1$  and  $V_2$  such that each edge of  $G$  connects a vertex of  $V_1$  to a vertex of  $V_2$ . Bipartite graphs are usually denoted by  $G = (V_1, V_2, E)$ . The degree of the vertices of a bipartite graph satisfied the following property, called 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  $E$  whose endpoints belong to  $V'$ . A graph  $G$  is called a *labeled graph* if its edges and/or vertices are assigned data of one kind or another. In particular, if each edge  $e$  of  $G$  is assigned a nonnegative number  $w(e)$  then  $w(e)$  is called the *weight* of  $e$ .

A particular type of graph that will be extensively used in this book are trees. A *tree* is a non-empty graph in which any two vertices are connected by a unique path. Given a tree, we will always designate a particular vertex, called the *root* of the tree, and direct each edge away from that root.

■ **Example 2.14** An equivalent definition of trees is provided by set theory. According to set theory a tree is a partially ordered set  $(T, <)$  such that for each  $t \in T$ , the set  $S = \{s \in T : s < t\}$  has an element that is smaller than every other element of  $S$  (*least element*). ■

Let  $T$  be a tree. If  $v$  is a vertex in  $T$  other than the root, the *parent* of  $v$  is the unique vertex  $u$  such that there is an edge connecting  $u$  to  $v$ . If  $u$  is the parent of  $v$ , then  $v$  is called a *child* of  $u$ . A *sibling* to a vertex  $v$  is any other vertex on the tree which has the same parent than  $v$ . The *ancestors* of a vertex are the vertices in the path from the root to this vertex, excluding the vertex itself and including the root. The *descendants* of a vertex  $v$  are those vertices that have  $v$  as an ancestor. A

vertex is called a *leaf* if it has no children. Vertices that have children are called *branches*. The *depth* of a vertex  $v$  is the length of the unique path from the root to  $v$ . The *height* of a tree is the maximum of the levels of its vertices.

■ **Example 2.15** The following applies to the tree depicted in Figure 2.3: the root is the vertex  $a$ ;  $c$  is a parent of  $d$  and  $d$  is a child of  $c$ ;  $d$  and  $g$  are siblings; the ancestors of  $d$  are  $a$  and  $c$ ; the descendants of  $c$  are  $e$ ,  $e$  and  $f$ ;  $b$ ,  $e$ ,  $f$  and  $g$  are leaf nodes;  $c$  and  $d$  are branches; the depth of  $d$  is 3; the height of the tree is 4. ■

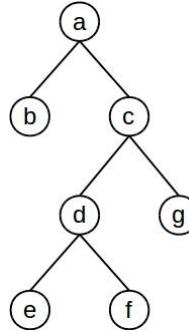


Figure 2.3: An Example of Tree

If  $v$  is a vertex in a tree, the *subtree* with  $v$  as root is the subgraph of the tree consisting of  $v$  and its descendants and all edges incident to these descendants. A tree is called an  *$k$ -ary tree* if every branch has not more than  $k$  children. The tree is called a *full  $k$ -ary tree* if every branch has exactly  $k$  children. An  $k$ -ary tree with  $k = 2$  is called a *binary tree*. A  $k$ -ary tree of height  $h$  is *balanced* if all its leaves are have a depth of  $h$  or  $h - 1$ .

■ **Example 2.16** A tree with  $n$  vertices has  $n - 1$  edges. A full  $k$ -ary tree with  $i$  branches contains  $m = ki + 1$  vertices. ■

TODO: Mention tree-traversals algorithms.

## 2.5 Counting Methods

### References

Two easy to read books on discrete mathematics that cover what is covered in this chapter and part of the the following chapters are [RK95] and [Epp10]. Good introductions to statistics and the theory of probability are [DeG+86] and [Spi+12].



### 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

*Probability* is a very difficult concept to grasp. Imagine we throw a dice and we want to calculate the probability that the number that appears is even. The dice has six possible outcomes, and since there are three even numbers, we say that the probability of having an even number is  $3/6$  or  $1/2$ . This what the *classical interpretation* of probability proposes: if we have an experiment in which all the possible (TODO: finite) outcomes are equally likely to happen, the probability of an event is the number of favourable cases divided by the total number of cases. The problem with this interpretations is that "equally likely" is essentially the same thing than "having the same probability", and so, it is a circular definition. An alternative approach to assign probabilities would be to apply the *principle of indifference*: in absense of any relevant evidence, all possible outcomes should have the same probability. The difficulty with this principle arises when there is evidence that not all cases are equal. For example, what happens if we know that the dice is loaded? How do we assign probabilities when not all the sides of the dice are equally likely?

The *frequentist interpretation* of probability proposes throwing the dice multiple times and compare the frequency of even numbers with the total number of throws. The general idea is to repeat the experiments a sufficiently large number of times under similar conditions and assing the relative frequency of each outcome as its probability. This interpretation has two main limitations. First of all, it is not clear how we should repeat an experiment under "similar conditions", since if we use exactly the same conditions the results of all the trials would be the same. The second is that it is not defined what a "large number of times" means (technically speaking, we should repeat the experiment an infinite number of times). From a practical point of view it is also very to apply the frequentist interpretation: some experiments cannot be repeated a large number of times, for example, what is the probability that a candidate wins an election?; probability is defined in terms of a succession of experiments, so we cannot compute the probability of an individual outcome; and we require that the relative frequency limit exists, that is not always the case, for example, in financial time series.

A third interpretation of the concept of probability, called *subjective interpretation*, proposes to assign to each event a probability based on our degree of belief: the more we believe that an event is true, the higher its probability. Of course, not all possible combinations of probabilities are valid, it is required that some rules of coherence must be satisfied. For example, if we are betting on the result of the dice thrown, an assignment of probabilities that guarantees that we will loose all of our money (what it is called a *dutch book*) does not satisfy the conditions of the subjective interpretation. It turns out that the conditions necessary and sufficient to guarantee a fair bet are the same conditions required by the axioms of probability introduced below. In this sense, we can assign to events whatever probabilities we want, as long as they are consistent with the axioms of probability. The problem with the subjective interpretation is that people have different degrees of belief. A solution to this problem is proposed by the *Bayesian interpretation* of probability: we start with a tentative assignment of probabilities, and as we get further evidence, we modify our degree of belief, or probability, accordingly; as we get more evidence, estimated probabilities will converge to the true probabilities. In any case, assigning probabilities to an infinite number of events is not, in general, human attainable.

Currently, the concept of probability is defined axiomatically (*axiomatic interpretation*), which means we give up in trying to define what a probability is, and instead we assume as true some of its properties. Intuitively, a probability should be a number between 0 and 1, where an event that cannot happen has a probability of zero, and an event that for sure will happen has a probability of

one. That fact that probability must be a number between 0 and 1 is more a social convention than a real mathematical requirement, since other ranges of numbers are equally good. We require some additional properties of probabilities. For example, if two events  $A$  and  $B$ , with probabilities  $P(A)$  and  $P(B)$  respectively, cannot happen at the same time, the probability that either  $A$  or  $B$  occurs should be  $P(A) + P(B)$ . If  $A$  and  $B$  can happen at the same time, but they are not related in any way, that is, they are independent events (whatever that means), we expect that the probability of the two events happening at the same time should be  $P(A)P(B)$ . Finally, we are also interested in the probability of  $A$  happening assuming that  $B$  has already happened, it should be the fraction of probability  $A$  that intersects with  $B$ . Probability, then, would be anything that satisfies these properties. The problem with the axiomatic interpretation is that too many things satisfy those requirements, like for example physical quantities like normalized mass or normalized volume.

Probability theory is about assigning a number to some events from a sample space. The word "event" is a little bit misleading in this context because it suggests that something happened, which is not always the case **TODO: provide an example**. However, to avoid confusion, we will keep calling events to what are essentially subsets.

**Definition 3.1.1** Let  $(\Omega, \mathcal{A})$  be a field over a non-empty discrete set.  $\Omega$  is called the *sample space*, its elements *outcomes*, and the elements of  $\mathcal{A}$  *events*. In particular,  $\Omega$  is called the *certain event* and the empty set  $\emptyset$  the *impossible event*.

As we saw in Section 2.1, given that  $(\Omega, \mathcal{A})$  is a field, we have that  $\Omega \in \mathcal{A}$  and that  $\emptyset \in \mathcal{A}$ . Moreover, the union of a finite collection of events is an event  $A_1 \cup A_2 \cup \dots \cup A_n \in \mathcal{A}$ , and that the intersection of a finite collection of events is also an event  $A_1 \cap A_2 \cap \dots \cap A_n \in \mathcal{A}$ .

As we have said in the introduction of this chapter, our main interest is in discrete mathematics, and so, we will mostly work with probabilities over discrete (finite or countably infinite) sets. An extension of the concept of probability to continuous sets requires the use of  $\sigma$ -algebras of sets instead of fields and the use of measure theory **TODO: explain in a footnote why this is the case**.

The standard axiomatization used in probability theory is called *Kolmogorov axioms*.

**Definition 3.1.2 (Kolmogorov's Axioms)** A *probability* is a real number  $P(A) \in \mathbb{R}$  assigned to each event  $A \in \mathcal{A}$  of the field  $(\Omega, \mathcal{A})$  that satisfy the following axioms:

**Axiom 1**  $P(A) \geq 0$ .

**Axiom 2**  $P(\Omega) = 1$ .

**Axiom 3** For every finite sequence of disjoint events  $A_1, A_2, \dots, A_n$  we have that  $P(\bigcup_{i=1}^n A_i) = \sum_{i=1}^n P(A_i)$ .

The triple  $(\Omega, \mathcal{A}, P)$  is called a *probability space*.

A problem with these axioms is that they cannot be reduced to first-order logic (See Appendix A), since real numbers cannot be described in terms of this logic. Also, no information is contained in the axioms about how probabilities can be assigned to events. **TODO: split the two ideas in two separate paragraphs and extend a little bit. Perhaps mention model theory**.

■ **Example 3.1** Let  $\Omega$  a sample space containing  $n$  elements equally probable. If  $A \subset \Omega$  is an event with  $d(A) = m$ , then we have that  $P(A) = m/n$ . ■

Let's prove some basic results about probabilities, starting by calculating the probability of the complement of an event, that is, the probability that the event does not happen.

**Proposition 3.1.1** For every event  $A$ ,  $P(A^c) = 1 - P(A)$

*Proof.* The sets  $A$  and  $A^c$  are disjoint and  $A \cup A^c = \Omega$ . Given Axiom 3 we have that  $P(A \cup A^c) = P(A) + P(A^c)$ , and given Axiom 2 we have that  $P(A \cup A^c) = P(\Omega) = 1$ , and so,  $P(A) + P(A^c) = 1$ . ■

As a direct consequence of previous proposition, we can derive the probability of the impossible event.

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

*Proof.* Given that  $P(\emptyset) = 1 - P(\Omega) = 0$  ■

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. ■

We have in place all the elements we need to prove that probabilities are numbers between zero and one.

**Proposition 3.1.4** For every event  $A$  we have that  $0 \leq P(A) \leq 1$ .

*Proof.* Based on Axiom 1 and given that  $A \subset \Omega$  and so  $P(A) \leq P(\Omega) = 1$  ■

Axiom 3 allows us to compute the probability of the union of disjoint events, but it says nothing about the case of non-disjoint events. Next proposition shows how to compute the probability of the union of events that are not disjoint.

**Proposition 3.1.5** For every two events  $A$  and  $B$ ,  $Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$ .

*Proof.* The union of sets  $A$  and  $B$  can be decomposed as the union of the two disjoint sets  $A \cup B = B \cup (A \cap B^c)$ , so given Axiom 3 we have that

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

In the same way, the set  $A$  can be decomposed as the union of the disjoint sets  $A = (A \cap B) \cup (A \cap B^c)$ , so that

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

Combining boths expression we get the desired result. ■

The following formula generalizes to the case of  $n$  events  $A_1, \dots, A_n$  **TODO: there is no need to use a named formula, since it is not reference bellow:**

$$\begin{aligned} P\left(\bigcup_{i=1}^n A_i\right) &= \sum_{i=1}^n P(A_i) - \sum_{i < j} P(A_i \cap A_j) + \sum_{i < j < k} P(A_i \cap A_j \cap A_k) - \\ &\quad - \sum_{i < j < k < l} P(A_i \cap A_j \cap A_k \cap A_l) + \dots + (-1)^{n+1} P(A_1 \cap A_2 \cap \dots \cap A_n) \end{aligned} \quad (3.1)$$

A probability function is a function that assigns to each possible event of a sample space its probability.

**Definition 3.1.3** Let  $(\Omega, \mathcal{A}, P)$  be a probability space. A *probability function* is a real value funcion  $f : \mathcal{A} \rightarrow [0, 1]$  such that for each  $A \in \mathcal{A}$  we have that  $f(A) = P(A)$ .

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

## 3.2 Conditional Probability

The concept of conditional probability plays a fundamental role in the area of statistical learning. Traditionally the conditional probability of event  $A$  given event  $B$  has been seen as the updated probability of  $A$  after we have learnt that  $B$  has occurred. However, this interpretation suggests that there is a temporal, even causal, relationship between events  $B$  and  $A$ , which is not necessarily true

**TODO: provide an example.**

Under the axiomatization of Kolmogorov, conditional probability is introduced as a definition. Some authors claim that conditional probability, being a central element in probability theory, should be a property that must be derived from the axioms. Of course, that would require to extend Definition 3.1.2 with additional properties. Unfortunately, there is no consensus among mathematicians and philosophers on how this extension should be carried out.

**Definition 3.2.1** Let  $A$  and  $B$  two events such that  $P(B) \neq 0$ . The *conditional probability* of  $A$  given  $B$ , denoted by  $P(A | B)$ , is defined as

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

Conditional probability is itself a probability, since it satisfies the axioms. Conditional probability  $P(A | B)$  is not defined if  $P(B) = 0$ .

The probability that two events will happen together (although not necessarily at the same time as we have seen **TODO: be sure we have seen it**), given their conditional probabilities, is  $P(A \cap B) = P(A | B)P(B)$  or  $P(A \cap B) = P(A \cap B)P(B)$ . The formula  $P(A \cap B) = P(A | B)P(B)$  provides, perhaps, a more intuitive interpretation of the concept of conditional probability. There are even some authors that propose that conditional probability should be defined using this interpretation instead of a quotient.

The generalization of this formula for the case of  $n$  events, called *multiplication rule*, is

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

The concept of events independence plays also a very important role in probability theory and statistical learning.

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

From an intuitive point of view, the events  $A$  and  $B$  are independent if observing that  $B$  has occurred does not alter the probability of  $A$ . This property can be derived from the definition of independence.

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

*Proof.* Assume that  $A$  and  $B$  are independent, that is  $P(A \cap B) = P(A)P(B)$ , then

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

Now let's assume that  $P(A | B) = P(A)$ . Given the multiplication rule we have that,

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

The same applies if we interchange  $A$  and  $B$ . ■

As it was the case of conditional probability, some authors claim that independence, being a fundamental concept in probability theory, should be derived from the axioms, not introduced as a definition.

The concept of independence can be generalized to the case of multiple events: the events  $A_1, \dots, A_n$  are independents (or mutually independent) if for every subset  $A_{i_1}, \dots, A_{i_j}$  composed by  $j$  events ( $j = 2, 3, \dots, n$ ), we have that  $P(A_{i_1} \cap \dots \cap A_{i_j}) = P(A_{i_1}) \dots P(A_{i_j})$ .

■ **Example 3.2** There is some confusion about the difference between mutually exclusive, or disjoint, events and independent events. If  $A$  and  $B$  are two mutually exclusive events, it does not make too much sense to compute the probability that  $A$  will happen given  $B$ , since if  $B$  happens,  $A$  cannot happen; in the same way that it does not make too much sense to talk about the conditional probability that  $A$  will happen given  $B$  if the probability of  $B$  is zero. However, since Definition 3.2.2 does not explicitly exclude the case of  $A$  and  $B$  being mutually exclusive, we have to conclude that two mutually exclusive events are independent if, and only if, one of them (or both) have a probability of zero. ■

A particular interesting case is when events  $A$  and  $B$  are not independent but they become independent if we know that some other even  $C$  has happened. **Explain why, and provide a reference to the section in which counterfactuals are introduced.**

■ **Definition 3.2.3** Let's  $A$ ,  $B$  and  $C$  events such that  $P(B \cap C) > 0$ .  $A$  and  $B$  are *conditionally independent* given  $C$  if  $P(A | B \cap C) = P(A | C)$ .

Next theorem introduces the Bayes' rule, which constitutes the foundations of a very important technique in statistical learning called Bayesian inference (see Section XX).

**Theorem 3.2.2** (Bayes' Theorem) Let's  $A$  and  $B$  two events such that  $P(B) \neq 0$ . Then we have that

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

The probability  $P(A)$  is called *prior probability*, and  $P(A | B)$  is called *posterior probability*.

*Proof.* From the definition of conditional probability we have that  $P(A | B) = P(A \cap B) / P(B)$  (given that  $P(B) \neq 0$ ) and that  $P(B | A) = P(A \cap B) / P(A)$  (given that  $P(A) \neq 0$ ). Solving for  $P(A \cap B)$  and substituting into the above expressions for  $P(A | B)$  gives us the theorem. ■

As we have seen in the proof, Bayes' theorem is a direct consequence of the definition of conditional probability (and we are not happy that conditional probability is a definiton). Its interpretation according to Bayesian inference is that it allows us to compute how our degree of belief about an event  $A$  (the prior probability  $P(A)$ ) changes when we gather additional evidence in the form of the occurrence of event  $B$  (and becoming the posterior probability  $P(A | B)$ ).

■ **Example 3.3** Let  $E$  a disease that affects to one of every one million persons,  $P(E) = 1 \times 10^{-6}$ , and let  $+$  a test designed to detect the disease that fails once every one thousand applications,  $P(+ | E) = 999/1000$ . We are interested in knowing the probability of having the disease if the test is positive  $P(E | +)$ . Applying Bayes' theorem we have that:

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

That is, although the test only fails once per thousand applications, it is still very unlikely we have the disease in the case of a positive result. This counterintuitive result is explained because the probability of failure of the test  $10^{-3}$  is much higher than the probability of having the disease

$10^{-6}$ . In practice we solve this problem by applying a second test to those who got a positive result, since the probability of having the disease after two positives is 0.5 (assuming that the successive repetitions of the test are independent). ■

Somewhere we should provide a second example in which the Bayes' theorem is meaningless.

Bayes theorem can be generalized to the case of multiple events: let's the events  $A_1, \dots, A_k$  ( $P(A_j) > 0$  for  $j = 1, \dots, k$ ) form a partition of the sample space  $\Omega$ , and let  $B$  be an event such that  $P(B) > 0$ , then for  $i = 1, \dots, k$  we have that

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

### 3.3 Random Variables

A random variable is a function that assigns a real number to the outcomes of an experiment. In this sense, a random variable acts as a numerical description of the experiment. Random variables are useful because they allow us to quantify the results of the experiments and study their properties from an analytical point of view. In fact, random variables are so useful that most of statisticians only think in terms of random variables, instead of probability spaces.

**Definition 3.3.1** Let  $(\Omega, \mathcal{A}, P)$  be a probability space. A *random variable* is a real-valued function  $X : \Omega \rightarrow \mathbb{R}$ .

The name random variable could be a little bit misleading. First, because random variables are not variables in the usual sense in algebra, they are functions. And second, because they are not random, what it is random is the experiment they describe. However, we will stay true to standard terminology.

Random variables are more useful when they describe properties of the experiments: if our sample space is composed by the students of a school, a random variable could assign to each student its height. Random variables allow us also to reassign the elements of the sample space into new events: if we throw two dices, a random variable could be the sum of the values of the dices. Be aware that we are free to assign a random variable to any sample space, even if that assignment does not make too much sense. For example, we could assign a number to each possible colour of a deck of cards, then pick up two cards at random, and sum their numbers; although the results do not have a meaningful interpretation, we could compute all sort of probabilities over them.

**Definition 3.3.2** Let  $X : \Omega \rightarrow \mathbb{R}$  be a random variable, and let  $C \subset \mathbb{R}$  be a subset such that  $\{\omega \in \Omega : X(\omega) \in C\}$  is an event. The probability that  $X$  belongs to  $C$ , denoted  $P(X \in C)$ , is given by  $P(X \in C) = P(\{\omega \in \Omega : X(\omega) \in C\})$ .

The probability distribution of a random variable describes how the probabilities are assigned to the values of the random variable.

**Definition 3.3.3** Let  $X : \Omega \rightarrow \mathbb{R}$  be a random variable. The *probability distribution* of the random variable  $X$  is the collection of probabilities  $P(X \in C)$  for all subsets  $C \subset \mathbb{R}$  such that  $\{\omega \in \Omega : X(\omega) \in C\}$  is an event.

The probability distribution of a random variable  $X$  defines a probability space over the real numbers (over the range of  $X$ ).

■ **Example 3.4** TODO: Example of how a probability distribution defines a probability space. ■

If  $X$  is a random variable whose range is discrete, it is said that  $X$  is a *discrete random variable* or that  $X$  has a *discrete distribution*. In this book we will consider only discrete random variables,

and we will use the name probability distribution to mean either the set of probabilities of a discrete probability space, or the distribution of a discrete random variable.

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 3.3.4** Let  $X$  be a random variable over a discrete probability space, and let  $\{x_1, x_2, \dots, x_i, \dots\}$  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, when dealing with discrete probability spaces  $(\Omega, \mathcal{A}, P)$  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 mass functions but to differ in significant ways; for instance, they may be independent, as next example shows.

■ **Example 3.5** Random variables can have the same distribution without being the same random variable. (this is also mentioned in the measures of centrality section). ■

Introduce the following proposition

**Proposition 3.3.1** Let  $X$  be a discrete random variable with p.f.  $f$ . If  $x$  is not one of the possible values of  $X$ , then  $f(x) = 0$ . Also, if  $\{x_1, x_2, \dots, x_i, \dots\}$  is the range of  $X$ , then  $\sum_{i=1}^{\infty} f(x_i) = 1$ .

*Proof.* TODO ■

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

Given the probability function of a random variable, we can derive its probability distribution.

**Proposition 3.3.2** If  $X$  has a discrete distribution, 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.* ■

Introduce the concept of Cumulative Distribution Function

**Definition 3.3.5** The distribution function or cumulative distribution function, abbreviated c.d.f.)  $F$  of a random variable  $X$  is the function  $F(x) = Pr(X \leq x)$   $-\infty < x < \infty$

The function  $F(x)$  is nondecreasing as  $x$  increases, that is, if  $x_1 < x_2$  then  $F(x_1) \leq F(x_2)$ . Also,  $\lim_{x \rightarrow -\infty} F(x) = 0$  and  $\lim_{x \rightarrow \infty} F(x) = 1$ .

Suppose that  $X$  has a discrete distribution with the p.f.  $f(x)$ .  $F(x)$  must have the following form:  $F(x)$  will have a jump of magnitude  $f(x_i)$  at each possible value  $x_i$  of  $X$ , and  $F(x)$  will be constant between every pair of successive jumps.

**Proposition 3.3.3** For every value  $x$   $Pr(X > x) = 1 - F(x)$

*Proof.* ■

**Proposition 3.3.4** For all values  $x_1$  and  $x_2$  such that  $x_1 < x_2$   $Pr(x_1 < X \leq x_2) = F(x_2) - F(x_1)$

*Proof.* ■

### Introduce the concept of random sample

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 3.3.6** 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  $(x_1, x_2, \dots, x_n) \in \mathcal{R}$ .

### 3.3.1 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 3.3.7** 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  $\Pr((X_1, X_2) \in C)$  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 3.3.8** 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 3.3.5** 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  $\Pr[(X, Y) \in C] = \sum_{(x,y) \in C} f(x, y)$

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 3.3.9** 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 3.3.10** 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 called the marginal p.f. or marginal p.d.f. of  $X$ .

**Proposition 3.3.6** 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 3.3.11** 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  $Pr(X \in A \text{ and } Y \in B) = Pr(X \in A)Pr(Y \in B)$

**Proposition 3.3.7** 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 3.3.8** 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 3.3.12** 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 3.3.9** 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 3.3.10** 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 3.3.11** 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 3.3.12** 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 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.

### 3.4.1 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 3.4.1** Let  $X$  be a discrete random variable whose probability function is  $f$ . The *mean* of  $X$ , denoted by  $E(X)$ , is defined as:

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

Of course, Definition 3.4.1 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 3.4.1** 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

$$\begin{aligned} E(a_1X_1 + \dots + a_nX_n + b) &= \sum_{x_1} \dots \sum_{x_n} (a_1x_1 + \dots + a_nx_n + b) f(x_1, \dots, x_n) = \\ &= \sum_{x_1} \dots \sum_{x_n} a_1x_1f(x_1, \dots, x_n) + \dots + \sum_{x_1} \dots \sum_{x_n} a_nx_nf(x_1, \dots, x_n) + \sum_{x_1} \dots \sum_{x_n} bf(x_1, \dots, x_n) = \\ &= \sum_{x_1} a_1x_1f(x_1) + \dots + \sum_{x_n} a_nx_nf(x_n) + b = a_1 \sum_{x_1} x_1f(x_1) + \dots + a_n \sum_{x_n} x_nf(x_n) + b = \\ &= a_1E(X_1) + \dots + a_nE(X_n) + b \quad (3.2) \end{aligned}$$

■

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

**Proposition 3.4.2** 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

$$\begin{aligned} E(X_1 \cdot \dots \cdot X_n) &= \sum_{x_1} \dots \sum_{x_n} (x_1 \cdot \dots \cdot x_n) f(x_1, \dots, x_n) = \\ &= \sum_{x_1} \dots \sum_{x_n} x_1 f(x_1, \dots, x_n) \cdot \dots \cdot \sum_{x_1} \dots \sum_{x_n} x_n f(x_1, \dots, x_n) = \\ &\quad \sum_{x_1} x_1 f(x_1) \cdot \dots \cdot \sum_{x_n} x_n f(x_n) = E(X_1) \cdot \dots \cdot E(X_n) \quad (3.3) \end{aligned}$$

■

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 3.4.2** 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 3.4.3** 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.

## 3.4.2 Measures of Dispersion

Definitions of the Variance and the Standard Deviation

**Definition 3.4.4** 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  $\sigma$ , and the variance by  $\sigma^2$ . Variance depends only on the distribution.

**Proposition 3.4.3** 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  $\mu$ . 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 3.4.4** 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 3.6** The Variance of a Binomial Distribution

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

### 3.4.3 Measures of Statistical Relationship

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

Covariance

**Definition 3.4.5** 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  $Cov(X, Y)$  is defined as  $Cov(X, Y) = E[(X - \mu_X)(Y - \mu_Y)]$

if the expectation exists.

The covariance between  $X$  and  $Y$  is intended to measre 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 3.4.5** For all random variables  $X$  and  $Y$  such that  $\sigma_X^2 < \infty$  and  $\sigma_Y^2 < \infty$   $Cov(X, Y) = E(XY) - E(X)E(Y)$

*Proof.*

Correlation

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

**Definition 3.4.6** Let  $X$  and  $Y$  be random variables with finite variances  $\sigma_X^2$  and  $\sigma_Y^2$  respectively. Then the correlation of  $X$  and  $Y$ , which is denoted by  $\rho(X, Y)$ , is defined as follows  $\rho(X, Y) = \frac{Cov(X, Y)}{\sigma_X \sigma_Y}$

XX

**Definition 3.4.7** 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 3.4.6** Moreover  $-1 \leq \rho(X, Y) \leq 1$

*Proof.*

**Proposition 3.4.7** If  $X$  and  $Y$  are independent random variables with  $0 < \sigma_X^2 < \infty$  and  $0 < \sigma_Y^2 < \infty$  then  $Cov(X, Y) = \rho(X, Y) = 0$

*Proof.*

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

**Proposition 3.4.8** 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 3.4.9** 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 3.4.10** 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 3.4.11** 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 3.4.12** 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.*

## 3.5 Common Distributions

### 3.5.1 Uniform Distribution

**Definition 3.5.1** 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 3.5.1** 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

### 3.5.2 Bernoulli Distributions

■ **Example 3.7** 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  $p$ .

### 3.5.3 Binomial Distributions

■ **Example 3.8** 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 3.5.2** 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.

## 3.6 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.

### 3.6.1 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 3.6.1 — Markov's Inequality.** Let  $X$  be a nonnegative random variable, i.e.  $P(X \geq 0) = 1$  with mean  $\mu$ . 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  $\mu$  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 3.6.2 — Chebyshev's inequality.** Let  $X$  be a random variable with mean  $\mu$  and variance  $\sigma^2$ . 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 3.6.1** 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 3.6.3 — 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

$$X_n \xrightarrow{P} \mu$$

*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 convers, but that it increases in absolute value as  $n$  increases (see Example 3.9).

■ **Example 3.9** 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. ■

### 3.6.2 Central Limit Theorem

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

**Theorem 3.6.4 — Central Limit Theorem.** Let  $X_1, \dots, X_n$  be a random sample of size  $n$  from a distribution with mean  $\mu$  and a finite variance  $\sigma^2$ . 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 3.10** 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.



## 4. Computability

*Caminante, no hay camino,  
se hace camino al andar.<sup>1</sup>*

Antonio Machado

We start our review of the background required to understand the theory of nescience by providing a mathematical formalization for the concept of *computable procedure*. Intuitively, a computable procedure is a method consisting of a finite number of instructions that when they are applied to a problem, after a finite number of steps, produce the correct answer. The key point is that the instructions must be so clear and precise that any human can follow them without aid. Moreover, we could go one step further and require that the instructions should be so simple that they could be performed by a machine. In 1936, the British mathematician Alan Turing proposed a formal model for a family of hypothetical machines and claimed that for every computable procedure (in its intuitive sense) there exist a Turing machine that can compute it. The model was simple enough to allow precise mathematical analysis, but also highly versatile and powerful. Over the years there have been many alternative proposals to formalize the concept of computable procedure, some of them very complicated, but it turned out that all of those models were equivalent to the concept of Turing machine; i.e. they solve the same set of problems. Two notable examples of alternative definitions are the *lambda calculus* proposed by Alonzo Church and the *theory of recursive functions* developed by Kurt Gödel and Stephen Kleene. Furthermore, the *Church-Turing thesis* states that any possible formalization of the concept of computable procedure must be equivalent to a Turing machine (as long as they satisfy some minimum requirements, for example, that they can only perform a finite amount of work in a single step). The Church-Turing thesis implies that there exists an objective notion of computable procedure that it is independent of any particular formalization.

The concept of Turing machine, referring to mechanical devices built to solve individual problems, can be extended and made universal. That is, there exists a machine that can solve all computable problems. This machine, called Universal Turing machine, works by simulating the

---

<sup>1</sup>Wanderer, there is no road, the road is made by walking.

behaviour of other particular machines, in the same way that current computers run algorithms written in programming languages. The concept of Universal Turing machine allows us to answer the very important question if there exists problems that are not computable. We will see that that the answer is yes, there are well defined problems that are beyond the capabilities of computers. And we will see that such kind of problems are more common than we could have imagined at first. This idea of uncomputable function will play a critical role in our theory of nescience.

Given the abstract nature of most of the entities under study in science, we have to resort to the concept of oracle Turing machine to properly formalize our theory. An oracle Turing machine is like a regular Turing machine but it has the capacity to query an external, not known how it works, oracle. This oracle can deal with those problems that are beyond the capabilities of Turing machines, that is, it can solve uncomputable problems. Quoting Turing "we shall not go any further into the nature of this oracle apart from saying that it cannot be a machine."

Recently there has been a huge interest in the area of *computational complexity*, a discipline that deals with the analysis of the number of steps that it takes to a Turing machine to solve a particular problem as a function of the size of the input to the problem. The interest has been mostly motivated due to the extremely difficult, and still unsolved, question  $P \stackrel{?}{=} NP$ ; that is, if the class  $P$  of problems that can be solved in polynomial time is the same than the class  $NP$  of problems whose solutions can be verified in polynomial time. In this book we are interested not only in the epistemological question of which problems can be solved (effectively) given enough time and space, but also in those problems that can be solved in polynomial time (efficiently).

## 4.1 Turing Machines

A Turing machine is a extremely simplified model of a general purpose computer, but capable of solving any problem that real computers can solve. From an intuitive point of view, we could see the machine as composed of a head that operates on a two-way infinite striped tape of symbols; at every time step the machine reads the symbol under the head and decides whether to write a new symbol on the tape, move the head one square (left or right), or do both things; algorithms are implemented using an internal table of rules in the control head, and the actual input to the algorithm is codified on the tape; when the machine reaches the final state, the algorithm's output is on 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.

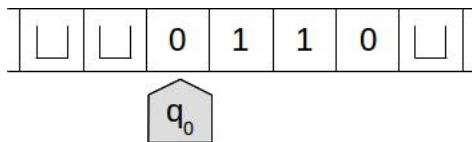


Figure 4.1: Turing Machine

**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*,
- $\Gamma$  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 implemented by the machine is given by the transition function  $\tau$ . This function specifies that given the current state of the machine and the tape symbol under the head, the machine will reach a new state, will write a new symbol on tape (or leave the old symbol unchanged), and will move the head to the left, to the right, or stay in place ( $L$ ,  $R$  or  $S$  respectively). After a finite, uniquely determined, succession of steps, the machine will reach the final state  $q_f$  and *halts* (the machine does not make any further move after halting). The output of the algorithm is the unique string of symbols  $s \in \Sigma^*$  that is left on the tape after halting. Eventually, some machines could iterate forever, never halting. If an undefined transition is reached, the machine will get stalled.

The input to the machine is a string of symbols, and we assume that in the initial state the head of the machine is located at the first symbol of the input string. If we want to solve a problem related to an object  $O$  other than a string, first we have to provide a method to encode that object using a string  $\langle O \rangle$ .

■ **Example 4.1** The following Turing machine solves the problem of adding two natural numbers: the set of states is  $Q = \{q_o, q_1, q_f\}$ , the set of tape symbols is  $\Gamma = \{0, 1, \sqcup\}$ , the set of input symbols is  $\Sigma = \{0, 1\}$ , and the transition function is given by the following table (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

Given the natural numbers  $n$  and  $m$ , the input string for this machine is  $n$  1's followed by a 0 and followed by  $m$  1's. The output is a string of  $n + m$  consecutive 1's. For example, if we want to add the numbers 2 and 3 the input string should be  $\sqcup 110111 \sqcup$ , and the machine output string would be  $\sqcup 11111 \sqcup$ . ■

A Turing machine can be also represented by a *state diagram*. A state diagram is similar to a labeled directed graph<sup>2</sup> where the vertices are the states of the machine, the edges represent a movement from one state to another state, and the edge labels shows the symbol under the head that yields to the new state, together with the symbol written down in the tape and the movement of the head. Under these conventions, the state diagram of the Turing machine of Example 4.1 is depicted in Figure 4.2.

It is a remarkable fact that small changes to the actual definition of Turing machine do not alter its power. That is, it is a highly robust definition. In Example 4.2 it is shown that adding more tapes to the machines does not increase the number of problems that can be solved using this model.

<sup>2</sup>In this particular case we allow loops and multiple edges coming out from vertices.

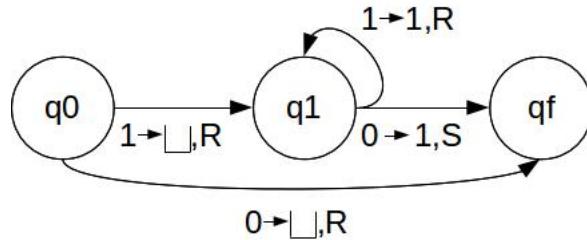


Figure 4.2: Example of Turing Machine

Similar proofs can be provided for the case of adding a finite storage to the control tape, allowing parallel processing with multiple control heads, and so on.

■ **Example 4.2** A *multitape Turing machine* is a Turing machine that has multiple heads with their corresponding tapes. At the starting configuration the input string is in tape 1 and all the others tapes are blank. The transition function of a multitape Turing machine is:

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

where  $k$  is the number of tapes. Multitape Turing machines are equivalent to regular Turing machines. We can prove this assertion by providing a mechanism in which a regular Turing machine can simulate the behavior of a multitape machine. In order to do that we have to encode the content of the multiple tapes into a single tape, introducing a new symbol that will act as tape separator, and to codify the location of the heads in all those tapes, by means of a new head location symbol. If the tape separation symbol is  $|$  and the head location symbol is  $h$ , a simulation tape of a 3-tapes machine could look something like  $\sqcup 01h00|000h1|h0101\sqcup$ . The operation of the regular machine just scans one by one the subtapes locating the head position and performing the required transition. If the computation of one subtape requires to write a new symbol beyond its limits, we have to shift the rest of the symbols to make room to the new symbol. Of course, the operation of the simulation will be much slower than the original multitape machine, but the set of problems that can be solved with the two types of machines is exactly the same. ■

Without any loss of generality, for the rest of this book we will assume that the set of input symbols  $\Sigma = \mathcal{B}$  and that the set of tape symbols  $\Gamma = \{0, 1, \sqcup\}$ .

**Definition 4.1.2** A *configuration* of a Turing machine  $T$  is the 3-tuple  $(q, s, i)$ , where  $q \in Q$  is an state of the machine,  $s \in \Gamma^+$  is a string with the contents of the tape (excluding the blank symbols), and  $1 \leq i \leq n$  is the index of the symbol  $s_i$  under the head, where  $s_1$  is the first non-blank symbol of the tape and  $n = l(s)$ .

Configurations allow us to losslessly describe the current state of a Turing machine. At any step of a computation we could stop the machine, write down its current configuration, and later on continue the computation at the same point where we left, just by reading its configuration.

**Definition 4.1.3** We say that configuration  $C = (q, s, i)$  yields 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

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

Given the concepts of configuration and configuration that yields a new configuration we can

formally define what we mean by computation.

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

Computations are deterministic, that is, given a Turing machine  $T$  and an input string  $s$ , the sequence of configurations is predetermined. If the machine  $T$  does not halt, or it gets stalled, on input  $s$ , we say that there was no computation.

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

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

■

The next theorem states that our intuitive notion of computable procedure is equivalent to the concept of Turing machine. This result is not a regular theorem, since it cannot be formally proved. Some authors call it *thesis* instead of theorem; Turing himself called it a *definition*.

**Theorem 4.1.1 — Turing's Thesis.** A procedure can be computed by a human being if, and only if, it can be computed by a Turing machine.

As we have mentioned in the introduction of this chapter, all the other alternative formalizations of the concept of computable procedure proposed so far are equivalent in expressive power to Turing machines.

## 4.2 Universal Turing Machines

In Section 4.1 we saw how to store the current state of a Turing machine so that the computation could be stopped and later on resumed. In Example 4.4 we are going to see a similar procedure, not to store the current state of the machine, but to save a full description of the machine itself. This procedure will allow us to enumerate, that is, to list all possible Turing machines. This enumeration will enable us to prove that there exists problems that are not solved by any Turing machine (see Section 4.3), and to introduce the very important concept of *Universal Turing Machine*.

■ **Example 4.4** In order to loosely describe a Turing machine we have 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  $q \in (Q \setminus \{q_f\})$ ,  $r \in Q$ ,  $s$  and  $t \in \Gamma$ , and  $a \in \{L, R, S\}$ . In this way, any Turing machine  $T$  will be 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 additional requirement that the first quintuple should refer to the initial state, and second one to the final state, that is  $q_1 = q_o$  and  $r_2 = q_f$ . A possible description of these quintuples would be to encode the elements of the set  $Q \cup \Gamma \cup \{L, R, S\}$  using a fixed length binary code (see Definition 5.1.5 for more information about these codes), so the quintuple  $(q, s, r, t, a)$  would be encoded as  $\langle q, s, r, t, a \rangle$ . The length of an encoded quintuple is  $5l$ , where  $l = \lceil \log(d(Q \cup \Gamma \cup \{L, R, S\})) \rceil$ . Using those conventions, the machine  $T$  would be 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 using 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.* If  $T_i$  is the  $i$ -th Turing machine encoded by the string  $\langle T_i \rangle$ , we can add an finite number of 0's to the end of this string, resulting in a new encoding index  $j$ , that encodes the same machine. ■

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

**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  $s \in \mathcal{B}$ .

Of course, we have to prove that such machine exists before we can use it. We could argue that a human being would be able to decode the machine  $T_i$  and simulate its behavior with the input string  $s$ , and then refer to Theorem 4.1.1. A better approach would be to explicitly build an universal Turing machine. However, describing one of these machines is out of the scope of this book. Instead, we refer to the reader to the references included at the end of the chapter.

### 4.3 Non-Computable Problems

Turing machines allow us to identify which problems can be solved using effective procedures, that is, the set of problems that can be worked out with computers. Although it might be a bit surprising, there exists many problems that cannot be solved using algorithms. These problems are beyond the capabilities of computers. We are not talking about problems like if a computer can be intelligent or if it can be self-aware, we refer to well defined mathematical problems. For example, the famous *halting problem* asks to find a computer program that, given any other program and an input string, decides if the program will eventually stop for that input string, or if it will keep running forever.

---

#### Algorithm 4.1 HALT function

---

```

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

---

**Theorem 4.3.1 — Halting Problem.** Define the HALT as in Algorithm 4.1. It does not exists 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  $\text{HALT}$  exists, and define a new Turing machine  $TC$  such that  $TC(A) = 1$  if  $\text{HALT}(A, A) = 0$ , and  $TC(A)$  will never stop if  $\text{HALT}(A, A) = 1$ . Then the contradiction arises when we ask about the result of  $TC(TC)$ : if  $TC(TC)$  stops we have that  $\text{HALT}(TC, TC) = 0$  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. ■

A very important consequence of the Halting Problem is that there exists more well-defined mathematical problems than computable solutions<sup>3</sup>. The Halting Problem has also important practical consequences in computer programming. For example, we cannot write a program that can guarantee that any other arbitrary program is bug-free, or that all infinite loops with conditional exits will eventually stop for all possible inputs.

Incomputability is not only about halting. Next example shows a well defined practical problem involving simple strings manipulation that can not 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  $\Sigma$ , where  $d(\Sigma) \geq 2$ , the *Post Correspondence Problem*, or PCP, asks to decide if there exists a sequence of  $K \geq 1$  indices  $(i_k)$ , where  $1 \leq i_k \leq n$  for all  $1 \leq k \leq K$ , such that  $\alpha_{i_1} \dots \alpha_{i_K} = \beta_{i_1} \dots \beta_{i_K}$ . For example, given the sequences  $(a, ab, bba)$  and  $(baa, aa, bb)$ , a solution to the problem would be  $\alpha_3 \alpha_2 \alpha_3 \alpha_1 = \beta_3 \beta_2 \beta_3 \beta_1$ . It does not exist an algorithm to solve PCP. As many proofs of incomputability, the proof proceed by showing that  $\text{HALT}$  can be reduced to PCP, that is, if PCP is decidable then the Halting problem should be decidable as well. We are not going to show the details of the proof in this section. For the interested reader, we refer to the references at the end of this chapter. ■

## 4.4 Computable Functions and Sets

Each Turing machine  $T$  defines a function  $f_T : \mathcal{B}^* \rightarrow \mathcal{B}^*$  that for each input string  $s \in \mathcal{B}^*$  assigns the output string  $T(s) \in \mathcal{B}^*$ . This relation between Turing machines and functions allows us to introduce the concept of *computable function*.

■ **Definition 4.4.1** A function  $f : \mathcal{B}^* \rightarrow \mathcal{B}^*$  is *computable* is there exist a Turing machine  $T$  such that it defines the function  $f$ .

Computable functions are also called *recursive functions*. However, since we are not going to cover the theory of recursive functions in this book, we prefer the term computable functions.

■ **Example 4.6** The function that assigns to each pair of natural numbers  $x$  and  $y$  its sum  $x + y$  is a computable function, as it is shown in Example 4.1. In this particular case, we have transformed the original function  $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  into a new function  $g : \mathcal{B}^* \rightarrow \mathcal{B}^*$  by means of encoding natural numbers as strings of 1's, and the pair of numbers  $x, y$  into a single string  $\langle x, y \rangle$ . ■

Partial functions can be defined by Turing machines that do not halt on the undefined values of the function.

■ **Definition 4.4.2** A partial function  $f : \mathcal{B}^* \rightarrow \mathcal{B}^*$  is *partial computable* is there exist a Turing machine  $T$  such that it defines the function  $f$  for those values in which  $f$  is defined, and  $T$  does not halt for those values in which  $f$  is not defined.

■ **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  $x < y$ . ■

We can apply the same concepts of computable and partial computable to sets.

<sup>3</sup>It is still an open question if there exists non-computable problems in Nature.

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

■ **Example 4.8** The set of all Turing machines that halt on all inputs is not computable, as we have shown in Theorem 4.3.1, although it is computably enumerable. ■

## 4.5 Oracle Turing Machine

TODO: Introduce the concept

TODO: Provide a diagram

With the aid of the Oracle, a Turing machine could solve uncomputable problems, like the halting problem.

The oracle tape is a one-way unbounded, read-only tape that contains all the values of the characteristic function  $\chi_{\mathcal{O}}$ . We assume that it takes, for arbitrary  $w \in \Sigma^*$ , only one step to search the tape and return the value of  $\chi_{\mathcal{O}}(w)$ .

**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*,
- $\Gamma$  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*.

Properties:

\* The machine is independent of the oracle set \* Turing machines are a subset of oracle Turing machines

Talk about:

\* How to encode oracle Turing machines \* What means a function or set to be oracle computable

\*

## 4.6 Computational Complexity

Computational Complexity theory is an investigation of the time, memory, or other resources required for solving computational problems. 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.

TODO: Adapt this definition.

**Definition 4.6.1** Let  $M$  a deterministic Turing machine that halts on all inputs. The running time or time complexity of  $M$  is the function  $f : \mathbb{N} \rightarrow \mathbb{N}$ , where  $f(n)$  is the maximum number of steps that  $M$  uses in any input of length  $n$ . If  $f(n)$  is the running time of  $M$ , we say that  $M$  runs in time  $f(n)$  and that  $M$  is an  $f(n)$  time Turing machine.

It considers only the highest order term of the expression for the running time of the algorithm, disregarding both the coefficient of that term and any lower order terms.

Introduce polynomial bounds  $O(n^c)$

**TODO:** Adapt this definiton. Let  $t : \mathbb{N} \rightarrow \mathbb{R}^+$  be a function. Define the time complexity class,  $\text{TIME}(t(n))$ , to be the collection of all languages that are decidable by an  $O(t(n))$  time Turing machine.

**TODO:** Add the following as an example: Let  $t(n)$  be a function, where  $t(n) \geq n$ . Then every  $t(n)$  time multiple tape Turing machine has an equivalent  $O(t^2(n))$  time single-tape Turing machine.

Polynomial differences in running time are considered to be small, whereas exponential differences are considered to be large.

Adapt this definition:

**Definition 4.6.2**  $P$  is the class of languages that are decidable in polynomial time on a deterministic single-tape Turing machine  $P = \bigcup_k \text{TIME}(n^k)$ .

$P$  roughly corresponds to the class of problems that are realistically solvable on a computer.

Add the following example of problem in  $P$ :  $\text{PATH} = \{ \langle G, s, t \rangle \mid G \text{ is a directed graph that has directed path from } s \text{ to } t\}$

Adapt this definition:

**Definition 4.6.3** A verifier for a language  $A$  is an algorithm  $V$ , where  $A = \{w \mid V \text{ accepts } \langle w, c \rangle \text{ for some string } c\}$ .

We measure the time of a verier only in terms of the length of  $w$ , so a polynomial time verifier runs in polynomial time in the length of  $w$ . A language  $A$  is polynomially verifiable if it has a polynomial time verifier. The symbol  $c$  is called a certificate, or proof of membership in  $A$ .

**Definition 4.6.4**  $NP$  is the class of languages that have polynomial time verifiers.

**TODO:** Include an example.

$P$  is the class of languages for which membership can be decided quickly.  $NP$  is the class of languages for which membership can be verified quickly. The question of whether  $P = NP$  is one of the greatest unsolved problems in theoretical computer science and contemporary mathematics.

**TODO:** Not sure about the rest of this chapter:

**Definition 14.** A function  $f : \Sigma^* \rightarrow \Sigma^*$  is a polynomial time computable function if some polynomial time Turing machine  $M$  exists that halts with just  $f(w)$  on its tape, when started on any imput  $w$ .

When problem  $A$  reduces to problem  $B$ , a solution to  $B$  can be used to solve  $A$ .

**Definition 15.** Language  $A$  is polynomial time mapping reducible, or simply polynomial time reducible, to language  $B$ , written  $A \leq_P B$ , if a polynomial time computable function  $f : \Sigma^* \rightarrow \Sigma^*$  exists, where for every  $w \in A \iff f(w) \in B$

The function  $f$  is called the polynomial time reduction of  $A$  to  $B$ .

If one language is polynomial time reducible to a language already known to have a polynomial time solution, we obtain a polynomial time solution to the original language.

**Theorem 16.** If  $A \leq_P B$  and  $B \in P$ , then  $A \in P$ .

**Definition 17.** A language  $B$  is NP-complete if it satisfies two conditions:  $B$  is in  $NP$ , and every  $A$  in  $NP$  is polynomial time reducible to  $B$ .

**Theorem 18.** If  $B$  is NP-complete and  $B \in P$ , then  $P = NP$ .

**Theorem 19.** If  $B$  is NP-complete and  $B \leq_P C$  for  $C$  in  $NP$ , then  $C$  is NP-Complete.

## References

The original paper form Alan Turing where the concepts of Turing machine, universal Turing machine, and non-computable problems were introduced is [Tur36], 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 [Sip12], and a more advanced introductions in [Coo03] and [Soa16]. In [Fer09] we can find a description of the most important computability models proposed so far. The Post Correspondence Problem was introduced by Emil Post in [Pos46]; for the details of the proof sketched in Example 4.5 please refer to [Sip12].

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



## 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, or 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 it, usually by means of computing and eliminating statistical redundancy. 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 play 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 get when some event happens is the logarithm of the inverse of the probability of that event. In this sense, the theory assumes that information is equivalent to surprise: the more unlikely is an event, the more information we get 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 average word length smaller than the entropy of the source alphabet.

There exists 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 join 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 it is exactly a code, since different authors propose different definitions. Fortunately, the definition of prefix-free code, the kind of codes required by 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,  $\mathcal{X} = \mathcal{B}$ .

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,  $C(ab) = C(ba) = 000$ . ■

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 an alternative 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, the restriction to  $C^n$  must be injective for all  $n$ . Now let 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 symbols  $s_3 = s_1 s_2$  and  $s_4 = s_2 s_1$ , 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 singular. ■

■ **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  $abac$ . The unique decodability is achieved because the 0 symbol, that plays the role of a comma, separating code words. ■

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 5.2.3). Note that we prefer the name *prefix-free code* to the more standard name of *prefix code*, since the former fits more accurately to the concept it names.

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

■ **Example 5.3** The code  $C(a) = 0, C(b) = 10, C(c) = 110$  and  $C(d) = 1110$  is a prefix-free code. The 0 symbol also 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 guarantees what makes the code prefix-free. ■

Fortunately, prefix-free codes are also uniquely decodable, as next proposition proves.

**Proposition 5.1.2** Let  $C$  be a prefix-free code, then  $C$  is uniquely decodable.

*Proof.* Let  $C$  be a fixed length code,  $C^n$  its extension of order  $n$ , and  $r = r_1r_2\dots r_n$  and  $s = s_1s_2\dots s_n$  two source words such that  $r \neq s$  but  $C^n(r) = C^n(s)$ . Then, we have that  $C(r_1)\dots C(r_n) = C(s_1)\dots C(s_n)$ , or equivalently that  $r_1^1\dots r_1^{i_1}\dots r_n^1\dots r_n^{i_n} = s_1^1\dots s_1^{j_1}\dots s_n^1\dots s_n^{j_n}$ . If  $i_1 \leq j_1$  we have that  $r_1$  is a prefix of  $s_1$ , and if  $i_1 > j_1$  the  $s_1$  is a prefix of  $r_1$ , which is a contradiction with the fact that  $C$  is prefix-free. ■

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. For example, given the code described in Example 5.2, after receiving the sequence 011 the source symbol could be a *c* or a *d*. Prefix-free codes present this property, this is why some authors prefer to call them *instantaneous codes*.

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

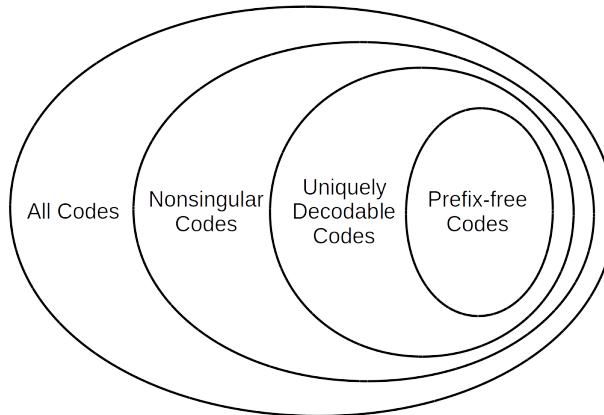


Figure 5.1: Classification of Codes

Another interesting type of codes are fixed length codes. We will use fixed length codes to compute the length of a text when we assume that there is any regularity we can use to compress the text.

**Definition 5.1.5** If all the code words of a code have the same length we say that the code is a *fixed length code*.

Fixed codes have the property of being prefix-free.

**Proposition 5.1.3** Let  $C$  be a fixed-length code, then  $C$  is prefix-free.

*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.

## 5.2 Kraft Inequality

The Kraft inequality provides a sufficient, and necessary, condition for the existence of a prefix-free code given a set of code words lengths. Among other things, Kraft inequality is important because if we take an exponential of the length of each codeword, the resulting set of values look like a probability mass function, that is, they sum less than or equal to one.

**Theorem 5.2.1 — Kraft Inequality.** Let  $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$  a set of lengths,  $l_i \in \mathbb{N}$ , 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 codeword. The prefix-free condition implies that nodes containing codewords cannot have descendants. An example of such a tree, for the code described in Example 5.3, is shown in Figure 5.2.

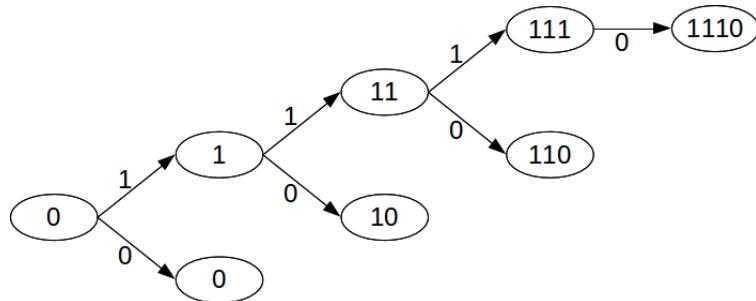


Figure 5.2: Prefix-free Tree

Let  $l_{max} = \max \{l_1, l_2, \dots, l_q\}$ , that is, the length of the longest codeword 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  $2^{l_{max}-l_i}$  leafs, since the code is prefix-free. Summing over all the codewords lengths, we have that the total number of pruned leafs must be less or equal than the maximum number of leafs, 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 codewords 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 codeword 2, and so on. Proceeding this way, we construct a prefix code with the specified lengths. ■

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

■ **Example 5.4** 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. ■

Kraft's inequality allows us to compare how efficient are the different codes available for the same source alphabet.

**Definition 5.2.1** Let  $C_1 : \mathcal{S} \rightarrow \mathcal{X}^+$  and  $C_2 : \mathcal{S} \rightarrow \mathcal{X}^+$  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 exist at least one  $s' \in \mathcal{S}$  such that  $l(C_1(s')) < l(C_2(s'))$ .

■ **Example 5.5** The code described in Example 5.4 is more efficient than the code of Example 5.3. Of course, the problem with the code described in Example 5.4 is that it is not prefix-free, but since it satisfy the Kraft's inequality, we know that there must exists another code with the same code word lengths that it 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.

**Definition 5.2.2** A code  $C$  is *complete* if there does not exists a code  $C'$  that is more efficient than  $C$ .

It turns out that the Kraft's inequality provides a very useful characterization of complete codes.

**Proposition 5.2.2** 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.* TODO: To be done ■

As we said above, in the theory of nescience we are mostly interested in prefix-fix codes. That might appear as a limitation, since it sounds more reasonable to use the more general class of uniquely decodable codes. However, such limitation does not exists, since, as next theorem proves, uniquely decodable codes also satisfy Kraft's inequality. That is, for any uniquely decodable code there exists a prefix-free code with exactly the same code word lengths. In the theory of nescience we are not interested in the codes themselves, but in code lengths.

**Theorem 5.2.3 — McMillan Inequality.** Let  $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$  a set of lengths,  $l_i \in \mathbb{N}$ , 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: To be done ■

### 5.3 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 5.3.1 — 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 B.1).

■ **Example 5.6** 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. ■

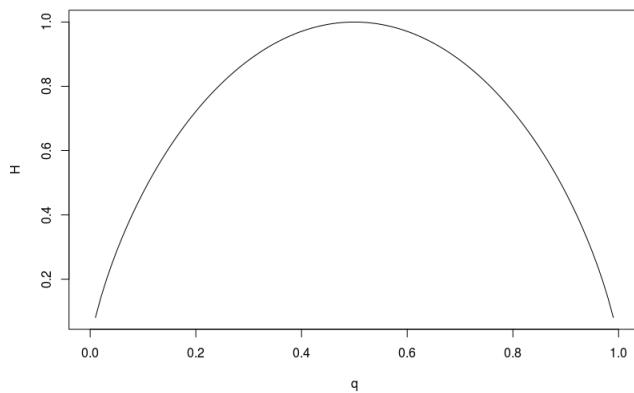


Figure 5.3: Binary Entropy Function

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 5.3.1** 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 B.1 we have that:

$$\log n - H(X) = \log e \sum_{i=1}^n p(a_i) \ln np(a_i)$$

And applying property B.2 (equalling  $x = 1/np(a_i)$ ):

$$\log n - H(X) \geq \log e \sum_{i=1}^n p(a_i) \left(1 - \frac{1}{np(a_i)}\right) \geq \log e \left(\sum_{i=1}^n p(a_i) - \frac{1}{n} \sum_{i=1}^n \frac{p(a_i)}{p(a_i)}\right) \geq 0$$

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 B.2 becomes an equality if, and only if,  $x = 1$ ). ■

■ **Example 5.7** 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 5.3.2** 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 5.3.2** 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 5.3.3** 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 5.3.3** 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 5.3.4** 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 Propositon 5.3.3 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 5.3.3. ■

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 5.3.5 — 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 5.3.4** 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 5.3.6** 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 5.3.7** 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 5.3.8** 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 5.8

## 5.4 Optimal Codes

TODO: Explain the concept of universal code. Give an example.

Lets fix  $\mathcal{S} = \{s_1, s_2, \dots, s_q\}$  a finite source alphabet, and  $P$  a probability distribution defined over the elements of  $S$ .

**Definition 5.4.1** The *expected length* for a code  $C$ , denoted by  $L_C$ , is defined as

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

where  $\mathcal{L} = \{l_1, l_2, \dots, l_q\}$  are the lengths of the code words of  $C$ . If  $C$  is clear from the context, we will denote  $L_C$  by  $L$ .

We are interested in finding a code  $C$  that minimizes the expected length  $L$  of the code words  $\mathcal{L}$  given the probability distribution  $P$ . This code  $C$  will allow us to compress the messages written with  $S$ , that is, to reduce the number of symbols required to write the messages.

**Definition 5.4.2** 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 definition provides a formal definition of this concept:

**Definition 5.4.3** 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.

**TODO: explain the relation between complete and compact codes**

Next theorem states that the entropy of the probability distribution  $P$  poses a limit to the average length of prefix-free codes.

**Theorem 5.4.1** 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 5.4.4** A probability distribution is called *D-adic* if each of the probabilities is equal to  $D^n$  for some  $n$ .

**Corollary 5.4.2** We have the equality in the theorem if, and only if, the distribution of  $X$  is D-adic.

*Proof.* TODO

**TODO: Explain how this relates to complete codes**

**TODO: Mention that optimal codes are random**

Mention that in practice we will non-integer codeword lengths. Show that, on average, the length of the encoded string will be less than 1 bit than using a code with codewords with integer lengths

## 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  $S = \{s_1, s_2, \dots, s_q\}$  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 5.2.3. The algorithm requires  $d(Q) - 1$  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,

**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

```

---

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 5.9** 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*

*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

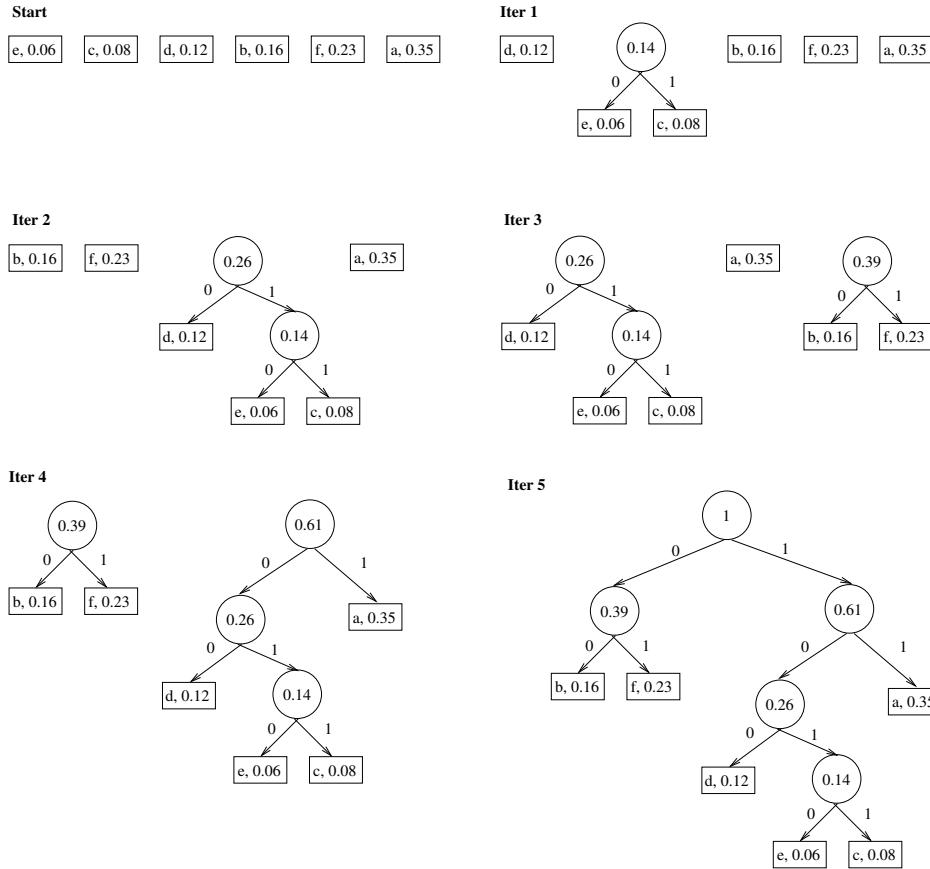


Figure 5.4: Huffman Algorithm

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 [Kra49]. The inequality was independently rediscovered and proved for the general case of uniquely decodable codes by Brockway McMillan in 1956 [McM56]. The proofs contained in this book have been adapted from [CT12].

The proof of proposition 5.3.1 has been adapted from [Abr63]. The proof of proposition 4.3.1 has been adapted from Abramson.

In the area of *digital signal processing* [GG12] 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 [Llo82] (closely related to the kmeans clustering algorithm [] 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 it was introduced the notion of complexity of a string based on the lengths of the code words of a prefix free code. We saw that this definition presents two limitations: the first one is that we need to know in advance the set of possible strings, and the second that it is necessary to define a priori a probability distribution over this set. It would be highly convenient if we can widen the set of strings until we cover all strings (that is,  $\mathcal{B}^*$ ), without requiring a probability distribution, so we provide an absolute notion of string complexity. Unfortunately, even if we are able to solve these problems, there exists an even more serious limitation when studying the complexity of strings using codes: some strings that we would expect to be classified as simple cannot be compressed at all. For example, the binary expansion of the  $\pi$  constant follows a uniform distribution over the set  $\{0, 1\}$ , and so, it cannot be compressed; but, on the other hand, it can be fully and effectively described by a very short mathematical formula. What it is required is an alternative definition of complexity of strings.

*Kolmogorov Complexity*, also known as *Algorithmic Information Theory*, provides a definition of complexity of a string that explicitly address these issues. Intuitively, the amount of information of a finite string is the length of the shortest computer program that is able to print the string. It does not require to know in advance the set of valid strings, nor its probability distribution. Moreover, objects like  $\pi$  are properly classified as having low complexity. We could say that Kolmogorov complexity provides a universal definition of amount of information that is very close to our intuitive idea. In order to compute the Kolmogorov complexity of a string, we have to agree upon a universal description method, or computer language, and a universal computer. We could rise the question if doing so the complexity of a string becomes a quantity that depends on the computer language selected. Fortunately, it can be shown that this is not the case, since all reasonable options (powerful enough) of computer languages provide the same description length, up to a fixed constant that depends on the selected languages, but now the string itself. Unfortunately, Kolmogorov complexity

introduces new problems, being the most important one that it is a non-computable quantity, and so, it must be approximated in practice.

At this point, we could ask ourselves if it is possible to compute the complexity of any object in general, not only strings. The answer is yes, at least in theory. Given an object  $x$  what we have to do is to provide an encoding method that describes the object using a string  $\langle x \rangle$ . That encoding method would be useful only if we are able to losslessly and effectively reconstruct the original objects given their descriptions. Unfortunately, it is not always possible to provide such descriptions, either because we are dealing with abstract objects (like most of the objects studied in mathematics), or because from a practical point of view it is not possible to reconstruct the object given its description (for example with living things<sup>1</sup>).

## 6.1 Strings Complexity

In Section 4.1 it was introduced the concept of Turing machine, an idealized model of computers, and we saw that Turing machines can be represented as partial computable functions  $T : \mathcal{B}^* \rightarrow \mathcal{B}^*$ , that assigns to each input string  $s \in \mathcal{B}^*$  the output string  $T(s) \in \mathcal{B}^*$  (Definition 4.4.1); we also introduced the concept of 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 it was introduced the concept of code, and in particular, the notion of prefix-free code (Definition 5.1.4), and we saw that this kind of codes present important properties (Theorem 5.2.3). Next definition merges the best of both worlds, Turing machines and prefix-free codes, and introduce 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  $U_v : \mathcal{B}^* \rightarrow \mathcal{B}^*$ ,  $U_v(x) = U(x, v)$  for all  $x \in \mathcal{B}^*$ .

Intuitively, what the above definition requires is 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 applying McMillan (Theorem 5.2.3) given a non-prefix-free program we could always find a prefix-free one that computes exactly 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 prefix-free universal Turing machine allows us to introduce a new definition of complexity of a string that is more in line with our intuitive idea of 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:

$$K(s) = \min_{p, v \in \mathcal{B}^*} \{l(p) + l(v) : U(p, v) = s\}$$

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 shortest description of a string  $s$  is given by two elements: a program 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 part.

<sup>1</sup>At least up to today it is not possible to recreate an animal given its DNA.

■ **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 the program 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 said in the preface of this chapter, Kolmogorov complexity would be useless if the complexity of the strings depends on the selected universal Turing machine. Next theorem proves that this is not the case, up to a constant that depends on the selected machines, but not on the strings themselves. In this way, we could say that Kolmogorov complexity is an inherent property of the strings themselves.

**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  $I(\langle U', p \rangle) = \log(I(\langle U' \rangle)) + 1 + I(\langle U' \rangle) + I(p) = K_{U'}(s) + C$ , where  $I(\langle U' \rangle)$  is the length of the machine  $U'$  using the encoding described in Section 4.2. ■

■ **Example 6.2** Select a universal programming language, for example Java, and an alternative language, for example Python. We could write an interpreter of Python in Java, that is, a Java program that as input gets a Python script and runs it. Then, we could just ask to Java to run this interpreter over the shortest Python script that prints a string. In this way, the complexity of a string  $s \in \mathcal{B}^*$  given the language Java  $C_J(s)$  would be less or equal than the complexity of the string given the language Phyton  $C_P(s)$  plus the length of the Phyton interpreter written in Java  $C_{J,P}$ , that is,  $C_J(s) \leq C_P(s) + C_{J,P}$ . Moreover, the lenght of the interpreter  $C_{J,P}$  does not depend on the string  $s$ . ■

Although we have proved that Kolmogorov complexity does not depends on the selected universal Turing machine, the size of the constant  $C$  could be a serious limitation in practical applications, since it prevents us to compute the complexity of short strings (the complexity of the string would be very small compared to the constant). This problem is explicitly addressed by the Minimum Description Length principle described in Chapter 7.

**Notation 6.1.** We denote by  $s^*$  the shortest program that prints the string  $s$  given the universal Turing machine  $U$ , that is  $s^* = \langle p, v \rangle$ ,  $U(s^*) = s$  and  $l(s^*) = K(s)$ . If there are more than one  $s^*$  programs satisfying the above 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 limitation that Kolmogorov complexity presents, another problem is that it is a non-computable quantity, that is, there is no algorithm that given a string computes the shortest program that prints that string.

**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 ■

In practice, what we do is to approximate the Kolmogorov complexity by the compressed version of the string, given a standard compression algorithm, for example, the Huffman algorithm described in Section 5.5.

## 6.2 Properties of Complexity

TODO: Section Pending!

**Proposition 6.2.1** 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.* TODO ■

The Kolmogorov complexity of a string cannot exceed its own length.

**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.* TODO ■

**Proposition 6.2.3** 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. ■

**Proposition 6.2.4** Let  $U$  and  $U'$  two universal Turing machines. Prove that there exists a constant  $C$  such that for each  $s \in \Sigma$  we have  $|K_U(s) - K_{U'}(s)| \leq C$ .

*Proof.* TODO ■

By  $K(x, y)$  we mean the length of the shortest program that prints the strings  $x$  and  $y$  in such a way that we can distinguish them.

**Proposition 6.2.5** There is a constant  $c$  such that for all  $x, y \in \mathcal{B}^*$  we have that  $K(x, y) \leq K(x) + K(y) + c$ .

*Proof.* TODO ■

**Proposition 6.2.6** There is a constant  $c$  such that for all  $x, y \in \mathcal{B}^*$  we have that  $K(x, y) \geq K(x) + c$  and  $K(x, y) \geq K(y) + c$ .

*Proof.* TODO ■

**Proposition 6.2.7** If  $f: A^* \rightarrow A^*$  is a computable bijection, then  $K(f(x)) < K(x) + O(1)$ .

*Proof.* TODO ■

### 6.3 Conditional Kolmogorov complexity

**TODO:** Section Pending!

Sometimes, the description of a string can be greatly reduced if we assume we know about another string. In this section we are going to introduce the concept of *conditional Kolmogorov complexity*, that is, the complexity of a string  $s$  when another string  $s'$  is given as input.

**Definition 6.3.1 — Conditional Kolmogorov Complexity.** Given the universal Turing machine  $U : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$  we define the *conditional Kolmogorov complexity* of a string  $s \in \Sigma^*$  given the string  $s' \in \Sigma^*$  as:

$$K(s|s') = \min_{p,v \in \Sigma^*} \{l(p) + l(v) : U(p, \langle v, s' \rangle) = s\}$$

As it was the case of the non-conditional Kolmogorov complexity, we have that the conditional complexity of a string  $s$  is a quantity that depends on the string itself and the provided string  $v$ , but not on the reference machine used for the computation.

**Exercise 6.1** Given  $U$  and  $U'$ , two universal Turing machines, prove that there exists a constant  $C_{U,U'}$  such that for each strings  $s$  and  $s' \in \Sigma^*$  we have that  $K_U(s|s') \leq K_{U'}(s|s') + C_{U,U'}$ . ■

**TODO:** Multiple conditional complexity

**Proposition 6.3.1** For all  $s \in \mathcal{B}^*$  we have that  $0 < K(s|v) < \infty$ .

*Proof.* TODO ■

**Proposition 6.3.2** There is a constant  $c$  such that for all  $s \in \mathcal{B}^*$  we have that  $K(s|s) = c$ .

*Proof.* TODO ■

The complexity of a string  $s$  does not increase when another string

**Proposition 6.3.3** There is a constant  $c$  such that for all  $s, s' \in \mathcal{B}^*$  we have that  $K(s|s') \leq K(s) + c$ .

*Proof.* TODO ■

**TODO:** Introduce this proposition.

**Proposition 6.3.4** For all  $s, s' \in \mathcal{B}^*$  we have that

$$K(s|s') \leq K(s) \leq K(s, s')$$

*Proof.* TODO ■

**TODO:** Introduce this proposition.

**Proposition 6.3.5** For all  $X, Y \in \mathcal{B}^*$  we have that

$$K(XY) = K(X) + K(Y | X)$$

*Proof.* TODO ■

## 6.4 Information Distance

**TODO: Section Pending!**

In this section we are going to introduce a universal measure of absolute information distance between individual objects. Intuitively, the information distance between two strings  $x$  and  $y$  would be the length of the shortest computer program that allow us to transform  $x$  into  $y$  and  $y$  into  $x$ . This notion of distance is universal in the sense that it contains all the other notions of computable distance as special cases.

We might be tempted to use as information distance the conditional Kolmogorov complexity of  $x$  given  $y$ ,  $K(x | y)$ , however conditional complexity is not suitable as a measure of information distance due to its assymetry, that is,  $K(\lambda | y)$  is always small, even in the case of  $y$  is not close to the empty string. Equally erroneously would be to use  $K(x | y) + K(y | x)$  since that would not take into account the redundancy existing between the information required to transform  $x$  into  $y$  and  $y$  into  $x$ .

**Definition 6.4.1** The *information distance* between two binary strings  $x$  and  $y$ , denoted by  $E(x, y)$ , is defined as

$$E_U(x, y) = \min\{l(p) : U(p, x) = y, U(p, y) = x\}$$

Say something about that information distance is up to a fixed universal Turing machine. Perhaps insists that it is not computable.

Introduce the following proposition.

**Proposition 6.4.1** Let  $x$  and  $y$  two binary strings, then we have that:

$$E_U(x, y) = \max\{K(x | y), K(y | x)\} + O(\log \max\{K(x | y), K(y | x)\})$$

*Proof.* TBD (based on the maximal overlap). ■

Introduce the concept of max distance as:

$$E(x, y) = \max\{K(x | y), K(y | x)\}$$

**■ Example 6.3** TODO: Give an example based on the bitwise exclusive or. ■

Introduce the following proposition.

**Proposition 6.4.2**  $E(x, y)$  is a metric.

*Proof.*

TODO: Introduce the concept of admisible information distance. Prove that  $E(x, y)$  is an admisible information distance. Prove that it is minimal for every admisable information distance. Explain this notion of universality.

Explain the problem of using non-normilized distance, perhaps with an example. "Normaliztion refers to the fact that the transformation description size must be seen in relation to the size of the participating objects"

TODO: Perhaps say something about the options of normalization, and why we choose max.

**Definition 6.4.2** The *normalized information distance* between two binary strings  $x$  and  $y$ , denoted by  $e(x, y)$ , is defined as:

$$e(x, y) = \frac{\max\{K(x | y), K(y | x)\}}{\max\{K(x), K(y)\}}$$

**Definition 6.4.3** The *normalized compression distance* between two strings  $x$  and  $y$ , given the compressor  $Z$ , and denoted by  $e_Z(x,y)$ , is defined as:

$$e_Z(x,y) = \frac{Z(xy) - \min\{Z(x), Z(y)\}}{\max\{Z(x), Z(y)\}}$$

Introduce the following proposition.

**Proposition 6.4.3** The normalized information distance  $e(x,y)$  takes values in the range  $[0, 1]$ .

*Proof.* TODO: Pending

Introduce the following proposition.

**Proposition 6.4.4** The normalized information distance  $e(x,y)$  is a metric, up to negligible errors.

*Proof.* TODO: Pending

TODO: Explain that NID is not computable in the general case, and how wonderfull would be to have a computable equivalent.

Introduce the following proposition.

**Proposition 6.4.5**

$$E(x,y) = \max\{K(x \mid y), K(y \mid x)\} = K(xy) - \min\{K(x), K(y)\} + O(\log K(xy))$$

*Proof.* TODO: Pending

Introduce the following definition.

**Definition 6.4.4** Normalize compression distance.

And its properties.

## 6.5 Incompressibility and Randomness

TODO: Section Pending!

It is easy to prove that the majority of strings cannot be compressed. For each  $n$  there are  $r^n$  string of length  $n$ , but only  $\sum$  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 6.5.1** 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 string 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!

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 [Sol64], Kolmogorov [Kol65] and Chaitin [Cha69].

In the Kolmogorov 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 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 16 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  $\theta$ , where  $\theta$  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  $\theta$ , of that distribution. When the parameter  $\theta$  is unknown, it is said that the distribution of the random variable is conditional to  $\theta$ .

We assume that the actual value of the unknown parameter  $\theta$  can be inferred, usually through a collection of data samples. The parameter  $\theta$  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  $\theta$  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  $\theta$ . The probability distribution of the parameter  $\theta$ , 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  $\Omega$ .  $\xi(\theta)$  is called the prior distribution because it is the distribution of the parameter  $\theta$  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  $\theta$ , and let  $X_1, \dots, X_n$  be  $n$  random variables. The conditional probability distribution of the parameter  $\theta$

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  $\theta$  is unknown and the prior p.d.f. or p.f. of  $\theta$  is  $\psi(\theta)$ . Then the posterior p.d.f. or p.f. of  $\theta$  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  $\theta$  taking values in a subset  $\Omega$  of the real line. An *estimator* of the parameter  $\theta$  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  $\theta$ .

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  $\theta$  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  $\theta$  on the set  $\Omega$ . 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  $\Theta$  be a family

of models with a discrete parameter  $\{\theta_1, \theta_2, \dots\}$  and with prior probabilities  $Pr(\theta_i)$ .  $\theta_i$  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  $\theta$ , 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  $\theta$ , and let  $\xi(\theta | \mathbf{x})$  denote the posterior p.d.f. of  $\theta$  on  $\Omega$ . 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  $\delta^*$  is called a *Bayes estimator* of  $\theta$ . Once  $\mathbf{X} = \mathbf{x}$  is observed,  $\delta^*(\mathbf{x})$  is called a *Bayes estimate* of  $\theta$ .

Another way to describe a Bayes estimator  $\delta^*$  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  $\theta$ .

■ **Example 7.3** The loss function  $L(\theta, a) = (\theta - a)^2$  is called squared error loss. Let  $\theta$  be a real-valued parameter. Suppose that the squared error loss function is used and that the posterior mean of  $\theta$ ,  $E(\theta | \mathbf{X})$ , is finite. Then, a Bayes estimator of  $\theta$  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  $\theta$ . ■

**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  $\theta$  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  $\Omega$ .

## 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  $\theta$ . 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  $\Theta$  be a family of models (probability distributions) with a discrete parameter  $\{\theta_1, \theta_2, \dots\}$  and with prior probabilities  $Pr(\theta_i)$ ,  $\theta_i$  could be a single scalar value or a vector value, and let  $x$  the data collected iid. For each model  $\theta$ , we assume the probability of getting data  $x$  given model  $\theta$ ,  $Pr(x | \theta)$ , is known. The *Maximum Likelihood Estimator*, or MLE, selects as best estimate the value of  $\theta$  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  $\epsilon$  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  $\varepsilon$  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 train MSE but 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 error bias of  $\hat{f}(x_0)$  and the variance of the error term  $\varepsilon$ :

$$E(y_0 - \hat{f}(x_0))^2 = Var(\hat{f}(x_0)) [Bias(\hat{f}(x_0))]^2 + Var(\varepsilon)$$

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,

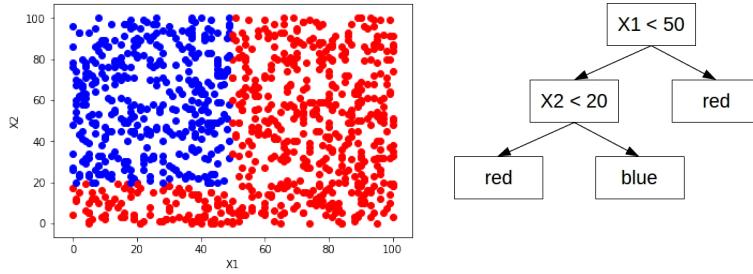


Table 7.1: Example of Decision Tree

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 particular country. Time series forecasting refers to the process

of building a model to predict future values of the series based on the previously observed values. The observed values could be continuous, discrete or even categorical.

Time series are analysed to understand the past and to predict the future [...] A time series analysis quantifies the main features in data and the random variation [...] When a variable is measured sequentially in time over or at a fixed interval, known as the sampling interval, the resulting data form a time series [...] we take a statistical approach in which the historical series are treated as realizations of sequences of random variables [...] referred to as a discrete-time stochastic process [...] The main features of many time series are trends and seasonal variations that can be modelled deterministically with mathematical functions of time. Another important feature of most time series is that observations close together in time tend to be correlated (serially dependent) [...] Once a good model is found and fitted to data, the analyst can use the model to forecast future values [...] Fitted models are also used as a basis for statistical tests. [...] Finally, a fitted statistical model provides a concise summary of the main characteristics of a time series. [...] The data may have been aggregated [...] or sampled [...] If data are sampled, the sampling interval must be short enough for the time series to provide a very close approximation to the original continuous signal when it is interpolated [...] 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. [...] Forecasting relies on extrapolation, and forecast are generally based on assumption that present trends continue. We cannot check this assumption in any empirical way. [...] outliers and erroneous values [...] need to be handled differently in the analysis and must not be included as observation when fitting a model to data. [...] it may be appropriate to consider robust methods of fitting models, which reduce the influence of outliers. [...] it is usually appropriate to remove trends and seasonal effects before comparing multiple series. [...] trend [...] change direction in unpredictable times [...] stochastic trend [...] two unrelated time series will be correlated if they both contain a trend.

**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  $\{x_1, x_2, \dots, x_n\}$ .

The elements  $x_i$  of the series correspond to values sampled at discrete times  $1, 2, \dots, n$ , and they can be continuous or categorical. In statistics, a time series is usually modeled as a sequence of  $n$  random variables, and a particular time series is a realization of the model. In this sense,  $\{x_t\}$  could represent a collection of random variables or the numerical observations of these random variables.

We represent a time series of length  $n$  by  $\{x_t : t = 1, \dots, n\} = \{x_1, x_2, \dots, x_n\}$ . It consists of  $n$  values sampled at discrete times  $1, 2, \dots, n$ . The notation will be abbreviated to  $\{x_t\}$  when the length  $n$  of the time series do not need to be specified. The time series model is a sequence of random variables, and the observed time series is considered a realization of the model.

The 'hat' notation will be used to represent a prediction or forecast. For example, with the series  $\{x_t : t = 1, \dots, n\}$ ,  $\hat{x}_{t+k|t}$  is a forecast made at time  $t$  for a future value at time  $t + k$ . A forecast is a predicted future value, and the number of time steps into the future is the lead time  $k$  [...]

### Trends and Seasons

Many time series ... and a repeating seasonal component.

**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**

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  $\mu$  and variance  $\sigma^2$ . The *autocorrelation* function, denoted by  $\rho$ , 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 careful 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 shaded areas in the correlogram used by `matplotlib` are computed. ■

Crosscorrelation measures ...

■ **Definition 7.2.10** Let  $\{\mathbf{x}_t\}$  and  $\{\mathbf{y}_t\}$  be time series with means  $\mu_x$  and  $\mu_y$  and variances  $\sigma_x^2$  and  $\sigma_y^2$ . The *crosscorrelation* function, denoted by  $\rho$ , 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  $\mathbf{x}_t$  be a time series. The structural decomposition of  $\mathbf{x}_t$  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  $\varepsilon_t$  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

$$\begin{aligned} y_t &= \mathbf{Z}_t \boldsymbol{\alpha}_t + \varepsilon_t \\ \boldsymbol{\alpha}_{t+1} &= \mathbf{T}_t \boldsymbol{\alpha}_t + \mathbf{R}_t \boldsymbol{\eta}_t \end{aligned}$$

where

$$\begin{aligned} \boldsymbol{\alpha}_t &\text{ explain} \\ \mathbf{Z}_t &\text{ explain} \\ \mathbf{T}_t &\text{ explain} \end{aligned}$$

$$\mathbf{y} = f(\mathbf{X}) + \varepsilon \tag{7.2}$$

measurement equation and transition equation

## 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 5.4.1). 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  $\theta$ .

**Definition 7.3.1** Let  $\Theta$  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 5.4). Note that  $\theta$  could be a single scalar or a vector of values. Moreover,  $\theta$  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  $\Theta$  and the functions  $f(X | \hat{\theta})$

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  $\mathbb{X} = \mathcal{B}^{100}$ ,  $\theta$  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  $(k, tita)$  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

In the area of *multiobjective optimization* we are interested in solving the following problem:

$$\begin{aligned} \text{minimize} \quad & \{f_1(x), f_2(x), \dots, f_k(x)\} \\ \text{subject to} \quad & x \in S \end{aligned}$$

where  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}$  are two or more objective *objective functions*, and the nonempty set  $S \subset \mathbb{R}^n$  is the *feasible region*, whose elements  $x = (x_1, x_2, \dots, x_n)$  are *decision vectors*. The image of the feasible region  $f(S) \subset \mathbb{R}^k$  is called *objective region*, and its elements  $z = (f_1(x), f_2(x), \dots, f_k(x))$  *objective vectors*.

In this book we will be dealing with nonlinear multiobjective optimization problems, where at least one of the objective functions is not linear. Objective functions can be also incommensurable, that is, measured in different units.

An objective vector is optimal if none of its components can be improved without deteriorating at least one of the others. In multiobjective optimization the objective functions are usually conflicting, and so, it does not exist a single solution that is optimal with respect to every objective function.

**Definition 7.5.1** A decision vector  $x \in S$  is Pareto optimal if there does not exist another decision vector  $y \in S$  such that  $f_i(y) \leq f_i(x)$  for all  $i = 1, \dots, k$  and  $f_j(y) < f_j(x)$  for at least one  $j$ . An objective vector  $z \in f(S)$  is Pareto optimal if there does not exist another objective vector  $w \in f(S)$  such that  $w_i \leq z_i$  for all  $i = 1, \dots, k$  and  $w_j < z_j$  for at least one  $j$ .

An objective vector is Pareto optimal if its corresponding decision vector is Pareto optimal.

[...] the Pareto optimal set (consisting of the Pareto optimal solutions) can be nonconvex and disconnected [...]

TODO: Add an example

[...] in practice, usually only one of these solutions is to be chosen [...] In multiobjective optimization, there are at least two equally important tasks: an optimization task for finding Pareto optimal solutions [...] and a decision-making task for choosing a single most preferred solution

[...] putting the preference articulation stage after optimization [...]

TODO: Introduce the following concepts: locally Pareto optimal, weak Pareto optimality

TODO: Introduce the following concepts: ideal vector (lower bound), nadir (upper bound), utopian

TODO: Introduce the following concept: Decision Maker [...] because Pareto optimal solutions cannot be ordered completely, we need extra preference information coming from a decision maker [...] explicit preference model [...]

[...] there is no DM and his preference information available and in those cases we must use so-called no-preference methods. Then, the task is to find some neutral compromise solution without any additional preference information. This means that instead of asking the DM for preference information, some assumptions are made about what a "reasonable" compromise could be

### 7.5.1 Trade-offs

[...] a trade-off is an exchange, that is, a loss in one aspect of the problem, in order to gain additional benefit in another aspect [...] a trade-off represents giving up in one of the objectives, which allows the improvement of another objective [...] how much must we give up of a certain objective in order to improve another one to a certain quantity

objective trade-off vs. subjective trade-off

**Definition 7.5.2** Let us consider two feasible solutions  $x^1$  and  $x^2$ , and the corresponding objective vectors  $f(x^1)$  and  $f(x^2)$ . Then, the ratio of change between  $f_i$  and  $f_j$  is denoted by  $T_{ij}$

### 7.5.2 Optimization Methods

Sometimes, there is no DM and his preference information available and in those cases we must use so-called no-preference methods. Then, the task is to find some neutral compromise solution without any additional preference information. This means that instead of asking the DM for preference information, some assumptions are made about what a "reasonable" compromise could be like.

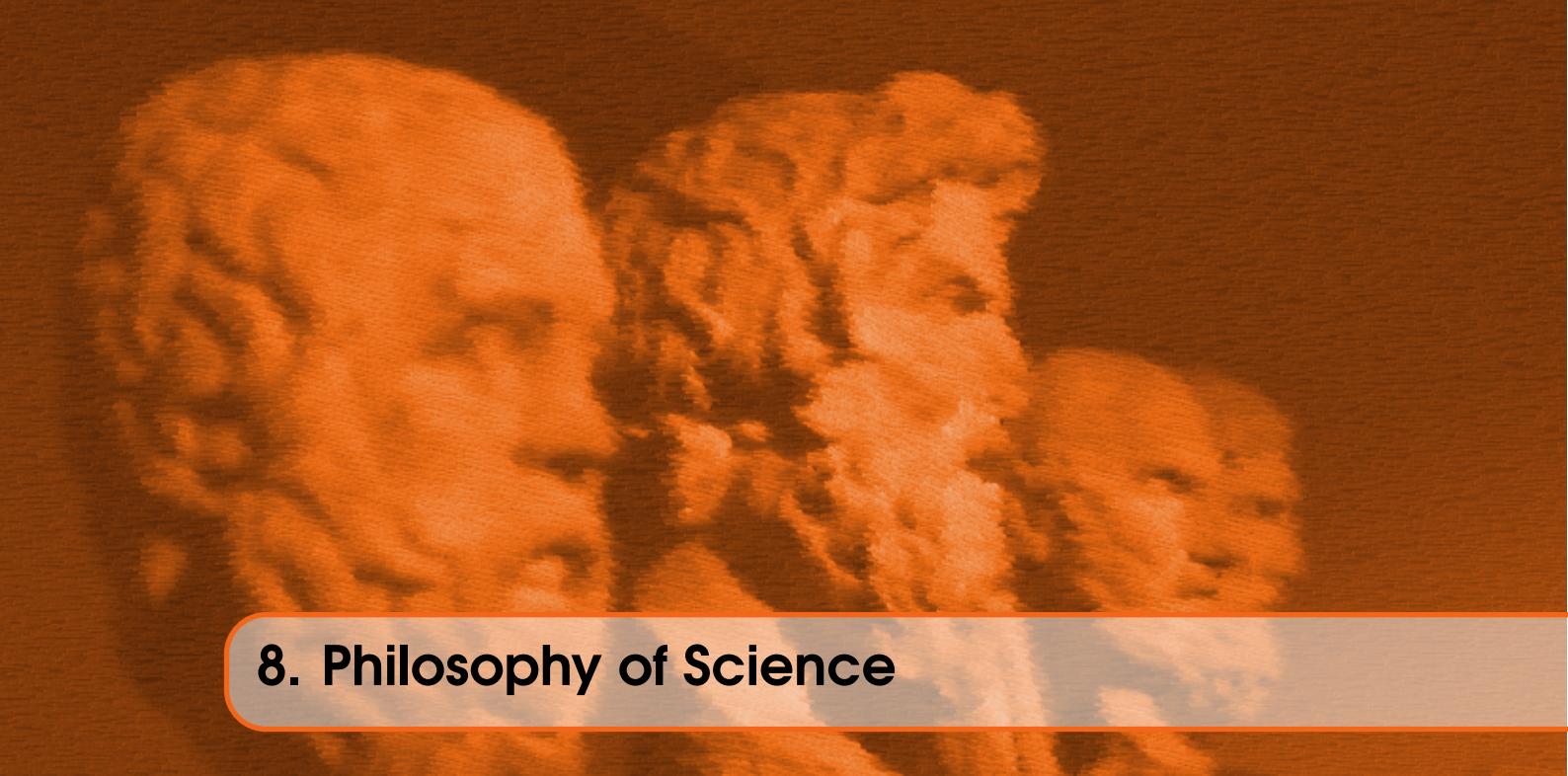
it is important that the multiobjective optimization method used can meet the following two needs: firstly, it must be able to cover, that is, find any Pareto optimal solution, and, secondly, generate only Pareto optimal solutions.

## 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 [WB68]. The book [Wal05], written by the same author, contains a detailed description of the principle.





## 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.

# Part 2: Foundations

## 9 Entities, Representations and Descriptions

125

- 9.1 Entities
- 9.2 Representations
- 9.3 Joint Representations
- 9.4 Descriptions
- 9.5 Descriptions for Joint Representations
- 9.6 Conditional Descriptions
- 9.7 Research Areas
- 9.8 References

10 Miscoding ..... 141

- 10.1 Miscoding
- 10.2 Miscoding of Joint Representations
- 10.3 Reducing Miscoding
- 10.4 Miscoding of Areas

11 Inaccuracy ..... 147

- 11.1 Inaccuracy
- 11.2 Conditional Inaccuracy
- 11.3 Inaccuracy of Areas

12 Surfeit ..... 151

- 12.1 Surfeit
- 12.2 Joint Surfeit
- 12.3 Conditional Surfeit
- 12.4 Surfeit of Areas

13 Nescience ..... 159

- 13.1 Nescience
- 13.2 Joint Nescience
- 13.3 Conditional Nescience
- 13.4 Nescience of Areas
- 13.5 Perfect Knowledge
- 13.6 Current Best Description
- 13.7 Nescience based on Datasets
- 13.8 Unknown Unknown

14 Interesting Questions ..... 167

- 14.1 Relevance
- 14.2 Applicability
- 14.3 Interesting Questions
- 14.4 New Research Topics
- 14.5 Classification of Research Areas

15 Advanced Properties ..... 177

- 15.1 The Axioms of Science
- 15.2 Scientific Method
- 15.3 The Inaccuracy - Surfeit Trade-off
- 15.4 Science vs. Pseudoscience
- 15.5 Graspness
- 15.6 Effort
- 15.7 Human Understanding
- 15.8 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 first step to quantitatively measure how much we do not know is to identify clearly the collection of research entities under study. The exact elements of this collection is something that depends on the particular application in which the theory of nescience is being used. Different applications require different collections of entities (mathematical objects, living things, human needs, etc.). Fortunately, the procedure to compute how much we do not know is the same in all cases.

The second step is to provide a method to encode the set of identified entities as strings of symbols, what we call representations. How to properly encode a research entity with symbols is a difficult, still unsolved, epistemological problem. The solution we propose in the theory of nescience is based in the concept of oracle Turing machine. How easy is to implement this solution in practice is something that depends on how abstract are the entities under study. For example, the collection of all abstract mathematical objects is a very difficult set to encode, and so, an approximation has to be found; the collection of all possible computer programs (given that our area of interest is software quality, see Chapter 17) is far easier to encode, since computer programs are strings themselves.

The final step, once we have found a way to properly encode the original set of entities as string-based representations, is to provide a description, as accurate and succinct as possible, of what we already know about those representations. In the theory of nescience we require that descriptions must be computable, that is, given a description, a computer should be able to fully reconstruct the original representation. A difficult problem that arise with descriptions is that they characterize representations, that is, the encoding of entities, not the entities themselves, and so, the quality of a description for an entity is conditional to the quality of the representation used.

In this chapter we will formalize all these concepts: entities, representations, descriptions, and many others. We will also see what we mean by a perfect description, how to compute the

combined representation of multiple entities, and the description of a representation assuming a perfect description of another one. The concept of research area, and its properties, will be also investigated.

## 9.1 Entities

What exactly is a research entity is a difficult, still unsolved, philosophical problem. Our approach to address this complex issue is eminently practical. Our theory starts by assuming there exists a non-empty collection of *entities* we would like to understand.

**Notation 9.1.** *We denote by  $\mathcal{E}$  the set of research entities in which we are interested.*

The exact elements that compose  $\mathcal{E}$  is something that depends on the particular domain in which the theory of nescience is being applied, but they usually corresponds to an area of knowledge. Examples of sets of entities could be: research elements in mathematics (abstract); the kingdom of animalia (living things); known and unknown human needs (abstract); all possible computer programs (strings), etc. Technically speaking  $\mathcal{E}$  is not a well defined set, since in general we cannot tell what is and what is not a member of this set.

The abstract nature of  $\mathcal{E}$  has some advantages, but also introduces important limitations. The main limitation is that our definition of nescience is, in general, a non-computable quantity, and so, it must be approximated in practice. The main advantage is that we can apply the new concepts and methods introduced in this book to other problems, not only to the discovery of new scientific knowledge.

In the theory of nescience we do not allow universal sets, that is, we cannot assume the existence of a set  $\xi$  that contains everything. The problem with universal sets is that they violate Cantor's theorem (see Example 9.1). Cantor's theorem states that the power set  $\mathcal{P}(\xi)$  composed by all the possible subsets of  $\xi$  has more elements than the original set  $\xi$ , and this is a contradiction with the fact that  $\xi$  contains everything. In the theory of nescience the set  $\mathcal{E}$  must be the set of something.

■ **Example 9.1 — Cantor's theorem.** The Cantor's theorem states that for every set  $A$  we have that  $d(A) < d(\mathcal{P}(A))$ . Let  $f : A \rightarrow \mathcal{P}(A)$  a function that maps every element of  $x \in A$  to the set  $\{x\} \in \mathcal{P}(A)$ ; clearly  $f$  is injective, and so,  $d(A) \leq d(\mathcal{P}(A))$ . In order to prove that the inequality is strict assume there exist a surjective function  $g : A \rightarrow \mathcal{P}(A)$ , and consider the set  $B = \{x \in A : x \notin g(x)\}$ . Since  $g$  is surjective there must exists an  $y \in A$  such that  $g(y) = B$ . However this rises the contraction  $y \in B \Leftrightarrow y \notin g(y) = B$ . Consequently, the function  $g$  does not exists, and so  $d(A) < d(\mathcal{P}(A))$ . ■

Not all possible sets are valid sets in the theory of nescience, because some sets lead to paradoxes. For example, the Russel paradox propose to consider the set  $R$  composed by all those sets that are not members of themselves; the paradox arises when we try to answer the question of whether  $R$  is a member of itself (see Example 9.2). In order to avoid these kind of problems, the theory of nescience is based on the Zermelo-Fraenkel set of axioms together with the axiom of choice (see Appendix B). Russell's paradox is due to an abuse of the set builder notation  $\{:\}$ . 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 contains all those sets  $u \in x$  that have property  $P$ ) only allows the use of this notation to construct sets that are subsets of already existing sets. A more general *axiom of comprehension* (if  $P$  is a property, the there exist a set  $y = \{u : P(u)\}$ ) would be required to allow sets like the one proposed by Russell's paradox. In the ZFC axioms, and in the theory of nescience, the axiom of comprehension is considered to be false.

■ **Example 9.2 — Russell's Paradox.** Let  $R$  be the set composed by all sets that are not members of themselves, that is,  $R = \{x : x \notin x\}$ . The contradiction arises when we ask if  $R$  is member of

itself. If  $R$  is not a member of itself, according to its own definition it should be; however if we say that  $R$  is a member of itself, the definition tell us that it should not be. Symbolically  $R \in R \Leftrightarrow R \notin R$ .

In the theory of nescience we do not deal with classical ontological questions, that is, about the classes of things that there exists in the world and that can be known. Also, we do not try to answer any kind of epistemological issues, like for example, how scientific knowledge is validated by appeal to evidence, or what it is the nature of that evidence.

Once we have selected a set  $\mathcal{E}$  of entities, the next step is to uniquely encode their elements as strings of symbols, otherwise it would not be possible to describe them (unless entities are strings of symbols themselves). How to properly perform this encoding is described in the next section.

## 9.2 Representations

How to represent the, possibly abstract, entities of a set  $\mathcal{E}$  is a difficult epistemological problem that has been the subject of research for more than two thousand years (see Section 8.2 for a short review of difficulties and open questions). In this book we describe a possible solution to this problem, and we will study its advantages and drawbacks. We propose to split the scientific representation problem in two complementary subproblems: how descriptions model representations, and how representations encode entities. In this sense, scientific descriptions are characterizations of entities by virtue of an indirect relation through representations (recall Figure 1.1 in the introduction of this book, reproduced again below for an easier reference). In this section we will focus on the encoding part, and Section 9.4 will be about descriptions.

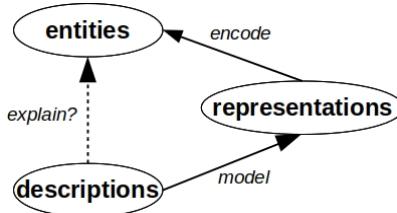


Figure 9.1: The Problem of Understanding.

In the discipline of Kolmogorov complexity, researchers solve the problem of representation by assuming that the set  $\mathcal{E}$  is well defined, countable, and that there exists a total encoding function  $f : \mathcal{E} \rightarrow \mathbb{N}$  from the set of entities to the set of natural numbers (a kind of Gödel numbering). In the theory of nescience we borrow this idea of encoding entities with numbers, or equivalently, strings of symbols. The main difference is that we address the problem of encoding an arbitrary set  $\mathcal{E}$  by turning it around, that is, by defining a total function  $\mathcal{O}_{\mathcal{E}} : \mathcal{S}^* \rightarrow \mathcal{E}$  from the well defined set  $\mathcal{S}^*$  of all possible finite strings from an alphabet  $\mathcal{S}$  to the, perhaps not countable and even not well defined, set  $\mathcal{E}$  of entities under study.

Our function  $\mathcal{O}_{\mathcal{E}}$ , that depends on the set  $\mathcal{E}$ , is a kind of oracle (inspired by the concept of oracle Turing machine from Definition 4.5.1) that is able to fully reconstruct an entity  $\mathcal{O}_{\mathcal{E}}(x) \in \mathcal{E}$  given its encoding string  $x \in \mathcal{S}^*$ , without requiring additional information. Of course, this oracle is a conceptual idea that cannot be built in the real world for the majority of the sets  $\mathcal{E}$ , but assuming its existence will allow us to explain and prove important properties of how the process of scientific discovery works.  $\mathcal{O}_{\mathcal{E}}$ , being an oracle and not a regular function, is limited in terms of the mathematical operations in which it can be used, as we will see below.

We assume that there exists a single, unique, physical world, that is independent of observers. We also assume that for some collections of entities  $\mathcal{E}$  it is free of logical contradiction to assume the existence of an oracle  $\mathcal{O}_{\mathcal{E}}$  (see the scientific representation problem in Section 8.2). Without

any loss of generality, for the rest of this book we will only consider binary strings as encoding of entities.

**Definition 9.2.1** Let  $\mathcal{E}$  be a set of entities. A *representation function* is an oracle  $\mathcal{O}_{\mathcal{E}} : \mathcal{B}^* \rightarrow \mathcal{E}$  from the elements of the set  $\mathcal{B}^*$  of all possible finite binary strings to the elements of the set  $\mathcal{E}$  of entities under study.

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).

■ **Example 9.3** Given a set of entities  $\mathcal{E}$ , if a representation oracle  $\mathcal{O}_{\mathcal{E}}$  exists, it is not unique. For example, a binary not oracle (transform the zeros of a binary string into ones and the ones into zeros) that assigns to each string  $x \in \mathcal{B}^*$  the entity  $\mathcal{O}_{\mathcal{E}}(\neg x)$  would be also a representation function.

We could have used an universal oracle machine instead of individual oracles. An universal oracle machine is a machine  $\mathcal{U}_{\mathcal{O}}$  that given as input the encoding of an oracle  $\mathcal{O}_{\mathcal{E}}$  and a string  $s$ , it computes  $\mathcal{U}_{\mathcal{O}}(\langle \mathcal{O}_{\mathcal{E}}, s \rangle) = \mathcal{O}_{\mathcal{E}}(s)$ . Universal machines make more sense when they are applied to the universal set  $\xi$  that contains everything. However, that would make the process of scientific discovery more difficult in practice. We prefer to work with sets of entities  $\mathcal{E}$  that correspond to particular areas of knowledge, and select one oracle  $\mathcal{O}_{\mathcal{E}}$  for each set  $\mathcal{E}$  (the best one according to our current knowledge).

■ **Example 9.4** When the topics under study are animals we could start using as encodings a detailed description of the body of those animals. In this case, the oracle would be an hypothetical machine that given its description is able to reproduce the original animal. As we get a better understanding of the biology of life we can consider to use an alternative encoding based on the DNAs of the animals. Both encodings are valid representations of the entities. ■

We use an oracle function instead of an oracle relation  $\mathcal{R}_{\mathcal{O}} \subset \mathcal{B}^* \times \mathcal{E}$ , because our aim is not about matching strings with their corresponding entities, but building an entity given its representation. The capacity of the oracle of reconstruct the original entities is what justifies the fact that we can formulate hypotheses about entities given their representations (surrogate reasoning). According to the theory of nescience, scientific research would be not only about figuring out how to properly encode entities, but also about discovering the inner workings of the decoding oracles.

The goal of encoding entities in the theory of nescience is different from that of Shannon's information theory, as Example 9.5 shows.

■ **Example 9.5** Consider a set  $\mathcal{E}$  composed by 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 one with the string "1". Although those strings allow us to uniquely identify each book in the set, they are not proper encodings in the sense of the theory of nescience. Information theory is about how to uniquely identify an object given a reference and it requires that both, sender and receiver agree about a mapping between references and objects. Meanwhile, in the theory of nescience we are interested in how to provide a representation that captures all the details and nuances of the original objects. For example, given the strings "0" and "1" it is not possible to make any hypothesis about the influence of Cervantes in the work of Shakespeare. ■

An alternative way to deal with the problem described in Example 9.5, where we used artificially short descriptions, would be to require the set  $\mathcal{E}$  to be infinite, as Kolmogorov complexity does (we

can not cheat an infinite number of times). However, even requiring the set of entities to be infinite, not every possible encoding schema can guarantee that the highly desirable property of surrogate reasoning is satisfied.

As it was shown in Example 9.3 and Example 9.4, the entities of  $\mathcal{E}$  might be encoded in multiple ways (problem of style). Different oracles will admit different encoding schemas. Some of those strings are better representations of an entity than others (standard of accuracy). Which representation is the best depends on the kind of questions we are interested to answer. In any case, our aim should be to use those representations that make the size of the oracle (the amount of inner knowledge we assume the oracle contains) as small as possible.

Recall that the oracle is a total function, that is, all possible strings must represent an entity, including the incomplete ones. The more information is missing from a representation, the harder would be for us to understand how the oracle works. We must be careful with those representations that do not capture all the details about the original entities, since the results of our analyses could present some bias. In Chapter 10 we will study in detail the problem of errors due to the miscoding of abstract entities, and we will see that not only representations are about entities, but also we require that entities should be about representations (requirement of directionality).

**■ Example 9.6** The data used in time of Ptolomeo about the position of celestial bodies along the year was a misencoding of the entity "position of celestial bodies". A better encoding was the data provided by Tycho Brahe. Today, we have even better encodings. ■

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  $\{r \in \mathcal{B}^* : \mathcal{O}_{\mathcal{E}}(r) = e\}$ .

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  $E$ ; 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.

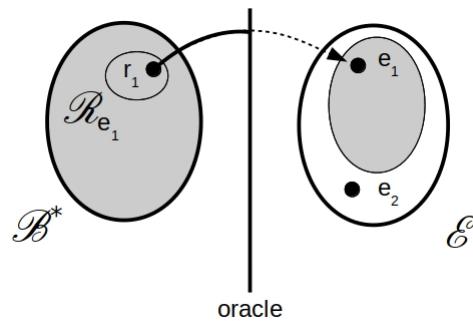


Figure 9.2: Encodings and Entities.

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  $\hat{\mathcal{R}}_e \subseteq \mathcal{B}^*$  of strings that we believe are close to the representations  $\mathcal{R}_e$  that encode the entity  $e$ . The elements that belong to  $\hat{\mathcal{R}}_e$  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  $\mathcal{E}$ , the more difficult will be to approximate them as strings in  $\hat{\mathcal{R}}_e$ .

■ **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. ■

A consequence of working with approximations (the set  $\hat{\mathcal{R}}_e$ ) instead of the true representations (the set  $\mathcal{R}_e$ ) is that some of the candidate strings currently in use might encode a different entity from what we were expecting.

■ **Example 9.9** 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. ■



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 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 9.3.1** Let  $s, t \in \mathcal{B}^*$  be two different representations. We call *joint representation* of  $s$  and  $t$  to the concatenation string  $st$ .

■ **Example 9.10** 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 9.3.1** 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  $\mathcal{O}_E$  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  $\mathcal{O}_E(s) \neq \mathcal{O}_E(t)$ . This process of joining representations from different entities will be very useful for the discovery of new research entities hitherto unknown (see Section 14.4 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 9.3.2** 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  $\lambda$  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,  $\mathcal{O}_E(st) \neq \mathcal{O}_E(ts)$ .

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 9.3.2** 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  $r_1 r_2 \dots r_n \in \mathcal{B}^*$  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.

## 9.4 Descriptions

So far, our aim with the strings of  $\mathcal{B}^*$  has been to provide an encoding, 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 9.11** In Example 9.10 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 smoking is not because they know and understand this large sample dataset, but because they know and understand the much simpler derived model "smoking increases the risk of lung cancer". ■

A description or model<sup>1</sup> 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 9.4.1 — 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. Descriptions that do not compress representations are useless, as we will see below in this section.

**Definition 9.4.2** 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:

$$\mathcal{D}_r = \{d \in \mathcal{D} : TM(a) = r\}.$$

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.

<sup>1</sup>In the theory of nescience use the words "description" and "model" interchangeably.

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 9.4.3** We call *description function*, denoted by  $\delta$ , 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 principle<sup>2</sup>, 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. Of course, what we know about the original entity is conditional to the quality of the representation.

**Definition 9.4.4** Given the set of descriptions  $\mathcal{D}_r$  of a representation  $r \in \mathcal{B}^*$ , let  $d_r^* \in \mathcal{D}_r$  be the shortest possible description of  $r$  using the standard shortlex ordering. We call  $d_r^*$  the *perfect description* of the representation  $r$ .

Unfortunately, the perfect description of a representation is in general not known and, as Proposition 9.4.1 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 the perfect one, that is, how much we do not know about a particular representation for an entity (see Chapter 12).

**Proposition 9.4.1** Given a representation  $r \in \mathcal{B}^*$ , we have that  $l(d_r^*) = K(r)$ .

*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  $\delta$  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 9.4.2** Let  $r \in \mathcal{B}^*$  be a representation,  $\delta$  and  $\dot{\delta}$  two different description functions, and  $d_r^*$  the perfect description of the representation  $r$  using  $\delta$  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 9.4.1 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 relevant<sup>3</sup>. For the rest of this book we will assume that  $\delta$  is fixed to a reference universal Turing machine. For example, in Section 15.1.2 we use as universal Turing machine the lambda calculus.

<sup>2</sup>The Occam's razor principle refers to the number of assumptions of an explanation, not to the length of the explanation itself.

<sup>3</sup>Do 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.

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 9.4.1 is that perfect descriptions must be incompressible, that is, *perfect knowledge implies randomness* (see Section 6.5).

**Corollary 9.4.3** 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 9.12** 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 a descriptions that is longer than the representation it describes.

**Definition 9.4.5** 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  $r$ .

■ **Example 9.13** 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 2.4 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 6.5 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 9.4.6** Given the set of descriptions  $\mathcal{D}_e$  of an entity  $e \in \mathcal{E}$ , let  $d_e^* \in \mathcal{D}_e$  be the shortest possible description of  $\mathcal{D}_e$  using the standard shortlex ordering. We call  $d_e^*$  the *perfect model* of entity  $e$ .

For each possible representation  $r$  of  $e$  we can compute its perfect description  $d_r^*$ . Then, the perfect description  $d_e^*$  for  $e$  would be the shortest of this collection of perfect descriptions of the representations.

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  $e$ .

## 9.5 Descriptions for Joint Representations

In Section 9.3 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 9.5.1** 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  $l(m_{ts}^*) \geq l(m_t^*)$  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 6.2.6. ■

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 9.5.2** 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 6.2.5. ■

An interpretation of Proposition 9.5.2 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 9.5.1 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 9.5.3** 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 6.2.1. ■

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 9.5.1, 9.5.2 and 9.5.3 can be generalized to any arbitrary, but finite, collection of representations  $t_1, t_2, \dots, t_n$ .

**Proposition 9.5.4** 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 9.5.1, 9.5.2 and 9.5.3 to individual pairs of topics  $i$  and  $j$ . ■

## 9.6 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 9.6.1** 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 9.6.2** 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 9.6.1** 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 9.6.1 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 9.6.3** 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 9.6.2** 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 6.3.3. ■

Next proposition shows the relation between the lengths of descriptions, joint descriptions and conditional descriptions.

**Proposition 9.6.3** 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 6.3.5. ■

As it was the case of joint descriptions, the concept of conditional description can be extended to finite collections of representations.

**Definition 9.6.4** 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 9.6.5** 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 9.6.2 and 9.6.3 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 9.6.4** 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 9.6.2 and 9.6.3 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 9.6.5** 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 9.6.2 and 9.6.3 to individual pairs of representations  $i$  and  $j$ . ■

## 9.7 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 9.7.1** 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 9.7.2** Let  $\mathcal{A} \subset \mathcal{E}$  be an area. We define the *known subset of the area*  $\hat{\mathcal{A}}$ , denoted by  $\hat{\mathcal{A}}$ , as the set composed by those entities  $e_1, e_2, \dots, e_n \in \mathcal{A}$  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 9.7.3** 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 9.7.4** 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 9.7.5** 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 9.4.5, if we are aware of the existence of a entity  $e \in A$ , that entity should be part of  $\hat{A}$ , even in the case we have not started yet to do research about that particular topic.

**Definition 9.7.6** Let  $A \subset \mathcal{E}$  be an area with known subset  $\hat{A}$ , and let  $d_{\hat{A}}^* \in \mathcal{M}_{\hat{A}}$  be the shortest possible description of  $A$ . We call  $d_{\hat{A}}^*$  the *perfect description of the area  $A$*  given the known subset  $\hat{A}$ , 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 9.7.1** Let  $A \subset \mathcal{E}$  be an area with known subset  $\hat{A} = \{e_1, e_2, \dots, e_n\}$ , then we have that  $l(d_{\hat{A}}^*) \leq l(d_{e_1}^*) + l(d_{e_2}^*) + \dots + l(d_{e_n}^*)$ .

*Proof.* Apply Proposition 9.5.4-ii. ■

Also, as it was proved in Proposition 9.5.4, 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 9.7.2** Let  $A, B \subset \mathcal{E}$  be two areas such that  $A \subset B$ , and let  $\hat{A}$  and  $\hat{B}$  be their known 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 9.7.3** Let  $A, B \subset \mathcal{E}$  be two areas with known 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 combine 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 5.3.5, we can provide a chain rule for the shortest length for a description of a research area.

**Proposition 9.7.4** Let  $A, B \subset \mathcal{E}$  be two areas with known 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 ■

## 9.8 References

**TODO:** Review this section.

For more information about Russell's paradox, Cantor theorem and universal sets refer, for example, to [Jec13]. 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 [Göd31]. A detailed description of the Berry

paradox from the point of view of computability can be found at [Cha95]. For a detailed account of the implications of Kolmogorov complexity being true up to a constant, please refer to [LV13]. That oracle machines are not mechanical was stated by Turing when he introduced the concept of oracle machine in [Tur39].

- 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
- Russel 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 most scientific disciplines, the set  $\mathcal{E}$  of entities under consideration will be composed by abstract elements, or other kind of complex objects, that cannot be studied directly. If we want to understand them we have to use an indirect method. As we have seen in the previous chapter, the approach proposed by the theory of nescience is to work with representations (i.e. strings of symbols) instead of using the original entities. Unfortunately, proceeding in this way introduce new problems: since we do not fully understand the elements of  $\mathcal{E}$ , otherwise we will not be doing research, the representations used probably will not be as complete and accurate as they should be. This limitation has serious implications, since an error in the representation of an entity will induce an error in the model we use to describe that entity. It is of utmost importance to characterize this type error, and to understand its implications.

Miscoding is a quantity that measures the error due to the use of bad encodings for entities. We propose a definition of miscoding based on the length of the shortest computer program that can print a correct representation given an incorrect one. Intuitively, miscoding quantifies the effort (measured as the length of a program) required to fix an incorrect representation. In practice, since the ideal representations of the entities are unknown, otherwise we would be using them, we cannot compute how far our current representation is from a perfect one. From a theoretical point of view we can take advantage of the oracle machine used to characterize the set  $\mathcal{E}$ , since that (abstract) machine knows the valid representations for all the entities. However, there are some limitations with respect to the kind of questions we can ask to the oracle that we have to take into account. For example, we cannot query the oracle about a particular entity for which we do not have a valid representation.

In this chapter we will formally introduce the concept of miscoding and study its properties. We will also see how miscoding behaves when dealing with joint representations, and what we can

do to decrease the miscoding. Finally, we will study how miscoding relates to research areas, and how it can be used to discover new research topics.

## 10.1 Miscoding

Miscoding refers to the fact that our current representation of an entity  $e \in \mathcal{E}$  might not be a valid one, that is, instead of working with a string  $r \in \mathcal{B}^*$  that perfectly encodes  $e$ , it might happen 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  $\mathcal{B}^*$ . Recall that the only thing we have to our disposal is an abstract oracle machine  $\mathcal{O}_{\mathcal{E}}$  that knows which strings represents each entity (see Section 9.2).

We start by distinguishing between valid representations, that is strings that contain all the relevant information required by the oracle to reconstruct an entity, and non-valid representations.

**Definition 10.1.1** Let  $\mathcal{E}$  be a collection of entities, and  $\mathcal{O}_{\mathcal{E}}$  and encoding function. We define the set of *valid* representations for  $\mathcal{E}$ , denoted by  $\mathcal{R}_{\mathcal{E}}^*$ , as the subset of representations that perfectly encode an entity of  $\mathcal{E}$ , according to  $\mathcal{O}_{\mathcal{E}}$ .

What it is a valid representation is something that depends on the particular oracle  $\mathcal{O}_{\mathcal{E}}$  selected, and in general, it is unknown. 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. In some knowledge areas, finding what it is exactly a valid representation, that is, finding how the oracle works internally, will be part of the scientific research activity.

A valid representation cannot include wrong information, or non-relevant information, even if the oracle is still able to reconstruct the original entity. As we have said, the oracle cannot query an external service to retrieve the information that is missing from the representation. In those cases, what the oracle does is to find the closest representation to the string provided that is valid, and assume that both strings describe the same entity.

We can define the set of valid representations of a single entity  $e$ , in the same way.

**Definition 10.1.2** 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,  $\mathcal{R}_e^* = \mathcal{R}_{\mathcal{E}}^* \cap \mathcal{R}_e$ .

It might happen that for some entities there is more than one valid representation. Those representations correspond to the different styles or ways to represent the entity (see Section 8.2). Moreover, different oracles will produce different sets of valid representations. It might also happen that the set of valid representation is empty, that is, there is no valid representation for the entity. This might happen even in case that the entity is knowable. In the theory of nescience, knowable means that we can derive some knowledge about an entity, but perfect knowledge is not guaranteed.

**Notation 10.1.** We denoted by  $r_e^*$  the fact that  $r$  is a valid representation of the entity  $e$ .

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 representation  $r$  from a valid representation  $r_e^*$ , for all  $r \in \mathcal{B}^*$  and all  $r_e^* \in \mathcal{R}_{\mathcal{E}}^*$ . Unfortunately, if we ask the oracle how far is a particular string  $r$  from perfectly encoding the entity in which we are interested, the oracle will require from us to specify the entity in which we are interested. And the only way we have to our disposal to tell the oracle which one is that entity is by means of using  $r_e^*$ , which, of course, we do not know. The work around we propose to solve this problem is to ask to the oracle how far is  $r$

from encoding *any* of the entities of  $\mathcal{E}$ . Something that can be answered by the oracle, at least in theory.

**Definition 10.1.3 — Miscoding.** Let  $r \in \mathcal{B}^*$  be a representation. We define the *miscoding* of  $r$ , denoted by  $\mu(r)$ , as:

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

Where the quantity  $\min_{r_e^* \in \mathcal{R}_e^*}^o$  has to be computed by the oracle. Intuitively, the more ignorant we are about an entity, the bigger will be the miscoding of our current representation, since a better understanding of that entity means that we should be able to provide an encoding closer to a perfect one. Recall that in our theory, all possible strings, even the most simplest ones, represent an entity.

Miscoding is computed using a two-way approach: we require the oracle to compute the length of the shortest computer program that can print the string  $r_e^*$  that encodes the closest entity given our representation  $r$ , and the other way around, that is, to compute the length of the shortest computer program that can print  $r$  given the string  $r_e^*$ . As expected, a good representation will include all the information required to reconstruct an entity, even if it is redundant, and it will not include wrong, or irrelevant, information. Miscoding is about including only relevant symbols (accurate), meanwhile surfeit (see Chapter 12) is about including only those symbols that are needed (concise).

In our definition of miscoding we have used a relative measure, instead of the absolute one, because besides to compare the miscoding of different encodings for the same entity, we are also interested in comparing the miscoding of different entities.

The miscoding of a representation  $r$  is always a number between 0 and 1.

**Proposition 10.1.1** We have that  $0 \leq \mu(r) \leq 1$  for all  $r \in \mathcal{B}^*$ .

*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 6.4.3. ■

Miscoding is equal to zero if, and only if, the representation  $r$  is one of the possible valid representation of an entity  $e$ .

**Proposition 10.1.2** Let  $r \in \mathcal{B}^*$  be a representation. We have that  $\mu(r) = 0$  if, and only if,  $r \in \mathcal{R}_e^*$ .

*Proof.* If  $r \in \mathcal{R}_e^*$  then there exists an entity  $e \in \mathcal{E}$  such that  $r = r_e^*$ , and so, we have that  $K(r | r_e^*) = 0$  and  $\mu(r) = 0$ . If  $\mu(r) = 0$  then there must exist an  $r_e^* \in \mathcal{R}_e^*$  such that  $K(r | r_e^*) = 0$ , which implies that  $r = r_e^*$  and  $r \in \mathcal{R}_e^*$ . ■

According to Proposition 10.1.2, if the miscoding of  $r$  is equal to 0 we can conclude that  $r$  perfectly encodes an entity  $e$ . The problem is that there is no way to know which one is the entity encoded by  $r$ . Of course, given our scientific intuition, we could guess the entity encoded, but from a mathematical point of view, we can not prove we are right. Moreover, with time and more research, it might happen that we change our mind about the nature of the encoded entity  $e$  (see Example 9.9).

An entity  $e \in \mathcal{E}$  can have multiple valid representations, given by the set  $\mathcal{R}_e^*$ . Fortunately, miscoding is a quantity that does not depend on the representation selected (see the problem of style in Section 8.2).

**Proposition 10.1.3** Let  $r_e^* \in \mathcal{R}_e^*$  be a valid representation, then we have that  $\mu(r_e^*) \leq \mu(r)$  for all  $r \in \mathcal{B}^*$ .

*Proof.* Given that  $\mu(r_e^*) = 0$  and  $\mu(r) \leq 0$  for all  $r \in \mathcal{B}^*$ . ■

Given an entity  $e$ , we have seen that all the valid representations that belong to  $\mathcal{R}_e^*$  are equally good from the point of view of miscoding, since all of them have a miscoding of 0. The question that arises is which one we should use in our research. From a practical point of view, we should select that representation that makes it easier to gather new knowledge about the original entity, that is, to derive models with low inaccuracy and surfeit.

■ **Example 10.1** Let  $\mathbf{X}_t$  be a time series composed  $m$  measurements  $x_1, \dots, x_m$  collected at fixed time intervals from a physical phenomena with a sinusoidal behavior, and assume we have trained a neural network  $nn$  that perfectly fits the data, that is, given a time  $t$  as input it returns  $x_t$ . Both representations have the same miscoding, that is  $\mu(\mathbf{X}_t) = \mu(nn)$ . An analysis of the cycles of the time series will allow us to discover that the best model for this particular physical phenomena seem to be a sine function, however, no currently known machine learning approach will arrive at the same conclusion having as input the architecture of the neural network (number of layers, sizes, and trained weights). Of course, the oracle is so smart that, in fact, it can do it. ■

## 10.2 Miscoding of Joint Representations

As we have seen in Section 9.3 if  $s, t \in \mathcal{B}^*$  are two representations, the concatenation  $st$  of those strings is also a representation, called joint representation. In this section we are interested in to study the miscoding of joint representations and their properties.

Given that the joint representation  $st$  is also a representation, its miscoding will be given by:

$$\mu(st) = \min_{r_e^* \in \mathcal{R}_e^*} \frac{\max\{K(st | r_e^*), K(r_e^* | st)\}}{\max\{K(st), K(r_e^*)\}}$$

The miscoding of a joint representation is a number between 0 and 1,  $0 \leq \mu(st) \leq 1$ , as a direct consequence of Proposition 10.1.1.

Adding more symbols to an incomplete representation, i.e. a representation with positive miscoding, is not a guarantee that the miscoding will decrease, since the added symbols might be wrong. In the same way, it is not always the case that the miscoding will increase, because adding relevant symbols will decrease miscoding of an incomplete representation. In symbols, given two arbitrary representations  $s, t \in \mathcal{B}^*$  none of the following relations will hold:  $\mu(ts) \geq \mu(t)$ ,  $\mu(ts) \leq \mu(t)$ ,  $\mu(ts) \geq \mu(s)$  nor  $\mu(ts) \leq \mu(s)$ .

It is not either the case that the miscoding of a joint representation is smaller than the sum of the miscoding of the individual representations, that is,  $\mu(st) \leq \mu(s) + \mu(t)$ . Not even in the case that both representations  $t$  and  $s$  encode the same entity, since it might happen that the joint presentation  $ts$  encodes a totally different entity.

Finally, since the operation of joining two representations is not commutative, it is not guaranteed that  $\mu(ts) = \mu(st)$ . Again, it might happen that the strings  $ts$  and  $st$  encode different entities.

We can extend the concept of miscoding of joint representations to any finite collection of representations. If  $r_1, r_2, \dots, r_n \in \mathcal{B}^*$  is a finite collection of representations, the miscoding of the joint representation  $r_1 r_2 \dots r_n$ , will be given as:

$$\mu(r_1 r_2 \dots r_n) = \min_{r_e^* \in \mathcal{R}_e^*} \frac{\max\{K(r_1 r_2 \dots r_n | r_e^*), K(r_e^* | r_1 r_2 \dots r_n)\}}{\max\{K(r_1 r_2 \dots r_n), K(r_e^*)\}}$$

As it was the case of joining two representations, we have that  $0 \leq \mu(r_1, r_2, \dots, r_n) \leq 1$ . But we cannot guarantee that  $\mu(r_1, r_2, \dots, r_n) \leq \mu(r_1) + \dots + \mu(r_n)$ , nor nothing about the miscoding of a permutation of the representations included in the joint representation.

We have seen that the concatenation of two or more representations of . 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.

**TODO:** Provide an example of how it can be applied in practice.

### 10.3 Reducing Miscoding

We have seen that a valid representation of an entity is a string that contains all the information needed by the oracle to reconstruct the selected entity, and only that information. If the representation is non-valid, that is, its miscoding is greater than zero, it could be due to the fact that some critical information is missing, or that it contains information that is non-relevant or even wrong. Accordingly, if the miscoding of a representation is non-zero, we could reduce the miscoding by adding the missing information, or by removing the wrong information from the representation. However, a priori there is no way to know if some information is missing, or some symbols are wrong. But, as next Theorem shows, it must be necessarily one of these two cases.

**Theorem 10.3.1** 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. ■

**TODO:** Provide a practical example.

### 10.4 Miscoding of Areas

**TODO:** Review and extend this section.

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 10.4.1** Let  $A \subset \mathcal{T}$  be an area with known subset  $\hat{A} = \{t_1, t_2, \dots, t_n\}$ . We define the *miscoding of the area* given the known subset  $d_{\hat{A}}$  as:

$$\mu(\hat{A}) = \min_{(t_{e_1}, t_{e_2}, \dots, t_{e_n}) \in \mathcal{T}_{\mathcal{E}}^n} \frac{K(\langle t_{e_1}, t_{e_2}, \dots, t_{e_n} \rangle | \langle t_1, t_2, \dots, t_n \rangle)}{K(\langle t_{e_1}, t_{e_2}, \dots, t_{e_n} \rangle)}$$

**TODO:** Say something about the concept of intradisciplinary and interdisciplinary in the context of miscoding and provide a reference to the chapter of computational creativity.

### References

**TODO:** Pending.





## 11. Inaccuracy

*A little inaccuracy sometimes saves tons of explanations.*

Saki

In Section 9.4 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  $r$  for an entity  $e$  is a Turing machine that, when interpreted by a universal Turing machine  $\delta$ , prints out the string  $r$ . However, since our knowledge about the entity  $e$  under study is usually incomplete, the description  $\delta(d)$  will transcribe a different string  $r'$ , which will be similar to  $r$ , but not equal. In this chapter we are going to study the error induced by bad models, i.e. how close the string  $r'$  is to the original string  $r$ . We refer to this type of error as the inaccuracy of the description  $d$ .

Inaccuracy is the second element we will use to characterize how well we understand a research entity. The intuition is that the more accurate our model is, the better we know the entity. From a formal point of view, we compute the inaccuracy of a description  $d$  as the normalized information distance between the original representation  $r$  and the representation  $r'$  produced by our description  $d$ . That is, inaccuracy is measured as the length of the shortest computer program that can fix the incorrect output of our model.

Inaccuracy compares the output of our description with the representation selected to encode the entity. However, this representation could be at the same time an incorrect one, as we have seen in the previous chapter. Inaccuracy is a concept that deals only with the description  $d$ , and does not take into account the fact that the representation  $r$  might have a positive miscoding. Furthermore, despite not requiring the use of the oracle, inaccuracy is a quantity that it is no computable for the general case, so it must be approximated in practice as we will see in Part III of this book.

In this chapter we will introduce formally the concept of inaccuracy and we will study its properties. We will also review how inaccuracy behaves when we use a conditional description of a representation compared to the unconditional one. And finally, we will extend the concept of inaccuracy from individual entities to research areas.

## 11.1 Inaccuracy

When studying an entity  $e \in \mathcal{E}$  through a representation  $r \in \mathcal{R}_e$ , it might happen that our candidate description  $d$  is not a valid description for  $r$ , that is,  $d \notin \mathcal{D}_r$  (see Definition 9.4.2). In that case, the universal Turing machine  $\delta$ , when given as input  $d$ , will print out a string  $r'$  different from the expected string  $r$ . Intuitively, we can say that  $d$  is an inaccurate description of the entity  $e$ . However, since descriptions describe entities indirectly through representations, our formal definition of the concept of inaccuracy has to be given with respect to the representations in use, not with respect to the original entities, and we should take into account that representations might be themselves wrong (something that has been already addressed with the concept of miscoding). Given the above considerations, we propose the following definition of the concept of inaccurate description.

**Definition 11.1.1** Let  $r \in \mathcal{B}^*$  be a representation, and  $d \in \mathcal{D}$  a description, with  $d = \langle TM, a \rangle$ . If  $TM(a) = r'$ , such that  $r \neq r'$ , we say that  $d$  is an *inaccurate* description for  $r$ .

Our candidate description  $d$  might not belong to the set of valid descriptions  $\mathcal{D}_r$  of  $r$  (positive inaccuracy), and the representation  $r$  might not belong to the set of valid  $\mathcal{R}_e^*$  representations of  $e$  (positive miscoding).

If our description is inaccurate, we would like to have a quantitative measure of how far we are from the right description. In terms of machines, a natural way to define this measure would be by means of computing how difficult is to transform the wrong representation  $r'$  produced by our description into the original representation  $r$ , that is, to compute the normalized information distance between  $r'$  and  $r$ .

**Definition 11.1.2 — Inaccuracy.** Let  $r \in \mathcal{B}^*$  be a representation, and  $d \in \mathcal{D}$  a description, with  $d = \langle TM, a \rangle$ . We define the *inaccuracy* of the description  $d$  for the representation  $r$ , denoted by  $i(d, r)$ , as:

$$i(d, r) = \frac{\max\{K(r | \delta(d)), K(\delta(d) | r)\}}{\max\{K(r), K(\delta(d))\}}$$

Having a relative measure of inaccuracy instead of an absolute one allow us to compare the inaccuracy of different descriptions for the same representation, and the inaccuracy of different descriptions for different representations.

Inaccuracy, as it was the case of miscoding (see Definition 10.1.3), is computed using a two-way approach: we compute the length of the shortest computer program that can print the correct representation  $r$  given the wrong one  $r'$ , and the other way around, that is, to compute the length of the shortest computer program that can print  $r'$  given the string  $r$ . That is, the representation generated by a valid description has to include all the information required to reconstruct an entity, but it cannot include wrong, or irrelevant, information.

■ **Example 11.1** Inaccuracy is about how difficult is to fix the output of a description, i.e. the output of computable model, not how difficult is to fix the description itself. If we have a dataset produced by system that can be perfectly described by a quadratic function, and we use as description a linear function, inaccuracy will compare the original quadratic dataset with the linear dataset predicted. Inaccuracy is not about how difficult is to transform the wrong linear model into the right quadratic one. In this sense, if the orginal dataset has 10 points, a ten degrees perfectly fitted polinomial would have also an inaccuracy of zero. Which model is the best between the zero inaccuracy quadratic and zero accuracy ten degrees polynomial is the subject of the surfeit metric (see Chapter 12). ■

Being based in the concept of Kolmogorov complexity, inaccuracy is a quantity that cannot be computed in practice for the general case, and so, it must be approximated. How to approximate the concept of inaccuracy is something that depends on the characteristics of the entities under study

and their representations.

The inaccuracy of a description is, conveniently, a number between 0 and 1, as next proposition shows.

**Proposition 11.1.1** We have that  $0 \leq i(d, r) \leq 1$  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 6.4.3. ■

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.

Inaccuracy is equal to zero if, and only if, the description  $d$  is one of the possible valid descriptions of the representation  $r$ .

**Proposition 11.1.2** Let  $d \in \mathcal{D}$  be a description for a representation  $r \in \mathcal{B}^*$ , we have that  $i(d, r) = 0$  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  $i(d, r) = 0$ . If  $i(d, r) = 0$  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  $d \in \mathcal{D}_r$ . ■

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:

$$i(d, rs) = \frac{\max\{K(rs | \delta(d)), K(\delta(d) | rs)\}}{\max\{K(rs), K(\Gamma(d))\}}$$

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  $0 \leq i(d, rs) \leq 1$ .

## 11.2 Conditional Inaccuracy

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 11.2.1** 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 11.2.2** 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 11.2.1** 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 6.4.3. ■

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 11.2.2** 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 11.2.1, 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  $0 \leq \iota(d_{rs|t}) \leq 1$ .

### 11.3 Inaccuracy of Areas

The concept of conditional inaccuracy can be extended to research areas in order to quantitative measure the amount of effort required to fix an inaccurate description of the area assuming some already existing background knowledge.

**Definition 11.3.1** Let  $A \subset \mathcal{T}$  be an area with known subset  $\hat{A} = \{r_1, r_2, \dots, r_n\}$ ,  $s \in \mathcal{B}^*$  a string, and  $d_{\hat{A}|s}$  a conditional description. We define the *inaccuracy of the area*  $d_{\hat{A}}$  as:

$$\iota(d_{\hat{A}|s}) = \frac{\max\{K(\langle r_1, r_2, \dots, r_n \rangle | \delta(\langle d, t \rangle)), K(\delta(\langle d, t \rangle) | \langle r_1, r_2, \dots, r_n \rangle)\}}{\max\{K(\langle r_1, r_2, \dots, r_n \rangle), K(\delta(\langle d, t \rangle))\}}$$

### References

A good introduction to the study of uncertainities (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.



## 12. Surfeit

*If you can't explain it simply,  
you don't understand it well enough.*  
Albert Einstein

**TODO:** Rewrite this introduction

Surfeit is a quantity that measures how redundant is a model of an entity. Intuitively, the more ignorant we are about an entity, the longer will be our current best model. Long models usually contain elements that are not needed, since a better understanding of the entity should allow us to remove all those unnecessary elements.

We define the surfeit of a entity's model as the difference between the length of this model and the length of the best possible model for that entity. In the theory of nescience we assume that the theoretical limit of what can be known about an entity, that is, its perfect model, is the shortest possible model that allows us to fully reconstruct the entity. **Again, clarify that entities are studied through the use of representations, and that surfeit is about representations**

The length of the shortest possible model of an entity is given by the Kolmogorov complexity of that entity. As we have seen, this quantity is not computable for the general case, that is, there is no algorithm that given the representation of an entity (a binary string) prints out its best possible model. Moreover, in practice, and given that our knowledge about entities is in general incomplete, we do not know the shortest possible model either. Surfeit is a quantity that has to be approximated in practice.

If we were able to come up with a perfect description of an entity, that model must be a random string, otherwise it would contain redundant elements that can be removed, and so, either it is not random or it is not perfect. By means of computing how far our current model is from being a random string we can estimate how far we are from having a perfect model.

Merge the following ideas into the introduction:

The key idea of this book is that *perfect knowledge implies randomness*. This is, in principle, a highly counterintuitive idea, since a lot of effort in science deals with the task to name, organize and classify our messy, chaotic, world. Even the kind of knowledge that explains how things work

requires a previous ordering and classification. Science, apparently, is anything but random. Yet, this is not the case.

The Theory of Nescience is based on the fact that randomness effectively imposes a limit on how much we can know about a particular topic. Far from being a handicap, the proper understanding of this absolute epistemological limitation opens new opportunities in science and technology, both, to solve open problems, and to discover new interesting research topics.

Our common understanding suggests that random strings do not make any sense, since this is what randomness is all about. On the contrary, we will see that random descriptions are the ones which contain the maximum amount of information in the less space possible.

## 12.1 Surfeit

Given the length of a description of a representation of an entity, and the length of its shortest possible description, we can introduce a relative measure of how much unneeded effort we are using to explain the entity when using that model. We call this quantity *surfeit*, and it will be part of our definition of nescience, that is, how much we do not know about that research entity.

**Definition 12.1.1 — Surfeit.** Given a representation  $r \in \mathcal{B}^*$ , and a valid description  $d \in \mathcal{D}_r^*$ , we define the *surfeit of the description d with respect to the representation r*, denoted by  $\sigma(d, r)$ , as

$$\sigma(d, r) = 1 - \frac{K(r)}{l(d)}$$

In our definition of surfeit we have used a relative measure instead of an absolute one (i.e.,  $l(d) - K(r)$ ), because besides to compare the surfeit of different descriptions for the same representation, we are also interested in to compare the surfeit of different entities. We prefer to use  $K(r)$  instead of the equivalent  $l(r^*)$  in order to be consistent with the definition of inaccuracy provided in Section 11.1.

Intuitively, the more ignorant we are about an entity, the longer will be our current best known model<sup>1</sup>, since a better understanding of that entity means that we should be able to remove all the redundant elements from that description.

A disclaimer that the descriptions may be non-valid

■ **Example 12.1 TODO:** Provide an example to clarify the concept. ■

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 valid descriptions, then we have that  $0 \leq \sigma(d, r) \leq 1$ .

*Proof.* Given that  $l(d) > 0$  and that  $K(r) > 0$ , since they are the lengths of non-empty strings, we only need to prove that  $l(d) \geq K(r)$ ; however  $l(d) < K(r)$  is a contradiction with the fact that  $K(r)$  is the length of the shortest possible Turing machine that prints out  $r$ . ■

Explain when it is the case that surfeit is zero, and 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 9.12). Such a description would not be redundant in the traditional sense, but it still will present some surfeit in the sense of the theory

<sup>1</sup>The concept of *current best model* will be introduced formally at Chapter 13.

of nescience. Moreover, as we have said above, it might happen that the description  $d$  we are considering does not perfectly describes 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 12.2** TODO: Provide an example to clarify the concept. ■

We have that the redundancy of a description is a number between 0 and 1.

**Proposition 12.1.2** 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 12.1.3** 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 12.1.3 applies to all possible description. Unfortunately, the proposition is true only when we deal with valid descriptions (from  $\mathcal{D}_r^*$ ), as Example 12.3 shows.

■ **Example 12.3** TODO: Provide an example. ■

## 12.2 Joint Surfeit

#### This section requires a full rewrite

The joint surfeit of two topics is given by our current understanding of both topics, and the shortest string that allows us to effectively reconstruct them.

**Definition 12.2.1** Let  $t, s \in \mathcal{T}$  be two different topics, and  $d_{t,s}$  a joint description. We define the *joint surfeit* of the description  $d_{t,s}$ , denoted by  $\sigma(d_{t,s})$ , as:

$$\sigma(d_{t,s}) = 1 - \frac{K(t, s)}{l(d_{t,s})}$$

As it was the case of the concept of surfeit, joint surfeit, being a relative measure, is again a number between 0 and 1.

**Proposition 12.2.1** We have that  $0 \leq \sigma(d_{t,s}) \leq 1$  for all  $t, s$  and all  $d_{t,s}$ .

*Proof.* Given that  $K(t,s) \leq l(d_{t,s})$  we have that  $\frac{K(t,s)}{l(d_{t,s})} \leq 1$  and so,  $1 - \frac{K(t,s)}{l(d_{t,s})} \geq 0$ . Also, since  $\frac{K(t,s)}{l(d_{t,s})} > 0$  (both quantities are positive integers), we have that  $1 - \frac{K(t,s)}{l(d_{t,s})} \leq 1$ . ■

When dealing with the joint surfeit of two topics, the order in which the topics are listed is not relevant.

**Proposition 12.2.2** We have that  $\sigma(d_{t,s}) = \sigma(d_{s,t})$  for all  $t,s \in \mathcal{T}$ .

*Proof.* Apply Propositions 6.2.1 and 9.5.3. ■

Our experience tell us that the more topics we include in a research, the less we understand the problem at hand. That is, the joint surfeit of two topics should be greater or equal than the surfeit of any of them isolated. Unfortunately, it is not the case that  $\sigma(d_{t,s}) \geq \sigma(d_t)$  for all combinations of  $d_{t,s}$  and  $d_t$ , since  $d_{t,s}$  and  $d_t$  are arbitrary descriptions of two different topics,  $ts$  and  $t$ . We have to wait until Chapter 13 when the concept of *current best description* will be introduced to formalize our intuition.

In the meantime we can prove a weaker result. In the case that the description  $d_{t,s}$  is just the concatenation of  $d_t$  and  $d_s$ , the case we need in our methodology for the discovery of interesting questions (see Chapter 14), we have that  $\sigma(d_{t,s}) \geq \sigma(d_t)$ .

**Proposition 12.2.3** Let  $t,s \in \mathcal{T}$  two topics, and  $ts$  the topic resulting as the concatenation of  $t$  and  $s$ , and let  $d_t$  and  $d_s$  descriptions of  $t$  and  $s$  respectively. If  $d_{t,s} = \langle d_t, d_s \rangle$  we have that  $\sigma(d_{t,s}) \geq \sigma(d_t)$  and  $\sigma(d_{t,s}) \geq \sigma(d_s)$ .

*Proof.* Given that Kolmogorov complexity is subadditive (see Proposition 6.2.5) and the fact that  $l(d_{t,s}) = l(d_t) + l(d_s)$ . ■

Intuitively, the joint surfeit of two topics in general should not be equal the sum of the individual surfeits, since it might happen that the topics partially overlap. However, as it was the case of Proposition 12.2.3, we have to wait until we define the concept of current best description to fully formalize this concept.

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 joint redundancy as a weaker version of the concept of joint surfeit.

**Definition 12.2.2** Let  $t,s \in \mathcal{T}$  be two different topics, and  $d_{t,s}$  a joint description. We define the *joint redundancy* of the description  $d_{t,s}$ , denoted by  $\rho(d_{t,s})$ , as:

$$\rho(d_{t,s}) = 1 - \frac{K(d_{t,s})}{l(d_{t,s})}$$

The joint redundancy satisfy the same properties that the joint surfeit.

**Proposition 12.2.4** For all  $t,s \in \mathcal{T}$  and all  $d_t \in \mathcal{D}_t, d_s \in \mathcal{D}_s$  we have:

- i  $0 \leq \rho(d_{t,s}) \leq 1$ ,
- ii  $\rho(d_{t,s}) = \rho(d_{s,t})$ .
- iii If  $d_{t,s} = \langle d_t, d_s \rangle$ , then  $\rho(d_{t,s}) \geq \rho(d_t)$ .

*Proof.* i Given that  $K(d_{t,s}) \leq l(d_{t,s})$  and that both quantities are positive integers.

ii Apply Propositions 6.2.1 and 9.5.3

- iii Given that Kolmogorov complexity is subadditive (see Proposition 6.2.5) and the fact that  $l(d_{t,s}) = l(d_t) + l(d_s)$ . ■

As it was the case of surfeit and redundancy, next Proposition shows that the joint surfeit is always greater or equal than the joint redundancy.

**Proposition 12.2.5** We have that  $\rho(d_t) \leq \sigma(d_t)$  for all  $t \in \mathcal{T}$  and  $d_t \in \mathcal{D}$ .

*Proof.* Apply the fact that  $K(d_t) \geq K(t)$  for all  $d_t$  and  $t$ . ■

Finally, we can extend our concepts of joint surface and joint redundancy to multiple, but fine, number of topics.

**Definition 12.2.3** Let  $t_1, t_2, \dots, t_n \in \mathcal{T}$  a finite collection of topics, and  $d_{t_1, t_2, \dots, t_n}$  a joint description. We define the *joint surfeit* of the description  $d_{t_1, t_2, \dots, t_n}$ , denoted by  $\sigma(d_{t_1, t_2, \dots, t_n})$ , as:

$$\sigma(d_{t_1, t_2, \dots, t_n}) = 1 - \frac{K(t_1, t_2, \dots, t_n)}{l(d_{t_1, t_2, \dots, t_n})}$$

And the *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 joint surface and joint redundancy apply to the case of multiple topics.

## 12.3 Conditional Surfeit

Rewrite this section, after Section 9.4 (conditional descriptions) is finished.

We are interested into study how the surfeit of a description for a representation is affected when some (perfect) background knowledge is given. That is, we want to know the surfeit of a conditional description for a representation, what we call *conditional surfeit*.

**Definition 12.3.1** Let  $t, s \in \mathcal{T}$  be two different topics, and let  $m_{t|s^*}$  a conditional model of  $t$  given  $s$ . We define the *conditional surfeit* of the model  $m_{t|s^*}$ , denoted by  $\sigma(m_{t|s^*})$ , as:

$$\sigma(m_{t|s^*}) = 1 - \frac{K(t | s^*)}{l(m_{t|s^*})}$$

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 12.3.1** We have that  $0 \leq \sigma(m_{t|s^*}) \leq 1$  for all  $t, s$  and  $m_{t|s^*}$ .

*Proof.* **TODO: adapt this proof.** Given that  $K(t, s) \leq l(m_{t,s})$  we have that  $\frac{K(t,s)}{l(m_{t,s})} \leq 1$  and so,  $1 - \frac{K(t,s)}{l(m_{t,s})} \geq 0$ . Also, since  $\frac{K(t,s)}{l(m_{t,s})} > 0$  (both quantities are positive integers), we have that  $1 - \frac{K(t,s)}{l(m_{t,s})} \leq 1$ . ■

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 13 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 12.3.2** Let  $t, s \in \mathcal{T}$  be two different topics, and let  $d_{t|s^*}$  any conditional description of  $t$  given  $s$ . We define the *conditional surfeit* of the description  $d_{t|s^*}$ , denoted by  $\sigma(d_{t|s^*})$ , as:

$$\rho(d_{t|s^*}) = 1 - \frac{K(d_{t|s^*})}{l(d_{t|s^*})}$$

Conditional surfeit is a relative measure, and so, a number between 0 and 1.

**Proposition 12.3.2** 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 12.3.3** 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.

## 12.4 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 12.4.1** 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  $K(d_{\hat{A}})$ . However, in the particular case of areas, we could have also problems with the quantity  $d_{\hat{A}}$ , since it requires to study the conditional descriptions of the topics included in the area.

**Definition 12.4.2** 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 [Abr63]).





## 13. 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

After the preparatory chapters XX, XX, XX, in this chapter we are ready to introduce the concept of nescience, and to study its main properties.

Nescience, on the contrary of what happens with the entropy of Shannon or the complexity of Kolmogorov, is not a measure of the quantity of information; instead, what it measures is the lack of information, that is, the unknown. According to the theory of nescience, how much we do not know about a research entity is given by three elements: miscoding, inaccuracy and surfeit. Miscoding measures how good is the representation of the original entity as a string of symbols that we can use in our research; inaccuracy measures how well our current best model describe that string of symbols; and sufait measures how well we understand the description itself, based on the amount of non necessary elements it contains. The problem is that these quantities are incompatible, in the sense that decreasing one of the could increase the other two. And so, we have to find a way to minimize the three quantities at the same time, that is, Nescience is a multi-objective minimization problem.

One of the most important consequences of our definition of nescience, based on the two metrics of error and redundancy, is that it divides the space composed by the research topics into two different areas. The first area is what we call the known unknown, that is, the set of topics that we do not fully understand, but that we are aware that we do not fully understand them. The second area is the unknown unknown, composed by those topics that have not been discovered yet. An important application of the theory of nescience is as a methodology for the discovery of what hides in the unknown unknown.

A second important consequence of the concept of nescience is the highly counterintuitive property that sometimes, for a certain class of topics, further research could be a counterproductive activity. That is, that more research we do, the less we know, since for these topics it is not possible

to increase our knowledge beyond a critical point, even if this point is far from being a perfect knowledge.

### 13.1 Nescience

Intuitively, how much we do not know about an entity should be based on the misscoding of the selected representation, and the surfeit and inaccuracy of the model used. MisCoding because ... Error because it says how close our description properly describes the topic, and redundancy because it tells us how unnecessarily effort are we spending with that particular description. Basically, what we are trying to achieve is solve the following minimization problem:

... trade-off among three conflicting objectives ...

According to the theory of nescience, the scientific method is about solving the following multi-objective minimization problem:

#### The Science Problem

$$\begin{aligned} \text{minimize} \quad & \{\mu(t), \iota(t, m), \sigma(m, t)\} \\ \text{subject to} \quad & (t, m) \in \mathcal{T} \times \mathcal{M} \end{aligned}$$

... no single solution exists that simultaneously optimizes each objective? ... there exists a (possibly infinite) number of Pareto optimal solutions ...

Probably, the most surprising fact about our characterization of the scientific method is that it does not involve the set  $\mathcal{E}$  of entities. Who could be that the scientific method does not require to know which are the entities under study? From a formal point of view we do not require to know the entity  $e \in \mathcal{E}$  under study because that would make the scientific method an ill-defined problem. For a practical point of view, this will allow us to provide practical approximations to the problem, since we are only dealing with string manipulations. And from a philosophical point of view, we consider that science is about the discovery of the unknown.

In any case the problem is still very difficult to solve, since the functions  $\iota$ ,  $\mu$ ,  $m$  are non-linear, non-convex and incommensurable.

The above optimization problem defines the following pareto optimal surface.

**Definition 13.1.1** An objective vector  $n^* \in \mathcal{T} \times \mathcal{M}$  is Pareto optimal if there does not exist another objective vector  $n \in \mathcal{T} \times \mathcal{M}$  such that  $\iota(t_2) \leq \iota(t_1)$ , ...

We could consider that descriptions lie in a two dimensional space where the axis are error and redundancy. The origin in this space would represent the perfect knowledge, that is, zero error and zero redundancy. Then, the nescience of a description would be the (Euclidean) distance between the point occupied by the description and the origin.

**Definition 13.1.2 — Nescience.** Let  $t \in \mathcal{T}$  a topic and  $m \in \mathcal{M}$  a model. We define the nescience of topic  $t$  given the model  $m$ , denoted by  $v(t, m)$ , as:

$$v(t, m) = \frac{3}{\mu(t)^{-1} + \iota(t, m)^{-1} + \sigma(m, t)^{-1}}$$

**TODO: Explain the intuition behind this concept.**

■ **Example 13.1** 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 13.1.2, 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  $d_1$ . ■

What Example 13.1 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).

**TODO:** Explain why not other metrics of distance.

**Proposition 13.1.1** 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  $t$ . ■

**TODO: Weak nescience**

**Approximation of Nescience**

Nescience is a theoretical concept that presents multiple issues when we try to use it in practice. In this section we are going to review these limitations, and will see how we can circumvent them.

The second problem with the theory of nescience is that we need to know the length of the shortest possible description of a topic. This poses a kind of contradiction: if we were aware of the shordest possible description, this one would be our best description, and the nescience would be equal to zero. The fact that our knowledge is incomplete implies that we do not know the shortest possible description of a topic. In order to deal with this problem, we will introduce a weak version of the concept of nescience. The weak nescience is based on the current best description alone, and it does not involves the best possible description at all:

$$\text{Weak Redundancy} = \frac{\text{length of best description} - \text{Kolmogorov complexity of best description}}{\text{Kolmogorov complexity of best description}}$$

Our definition of weak nescience is based on the Kolmogorov complexity of our best description. Kolmogorov complexity is a concept that also presents serious limitations when it is applied in practice. The first one is that, although the complexity does not change when we change the universal reference machine, this is true up to a constant (that depends on the selected machines, but not on the topic itself). In practice, the size of this constant could be very large in comparison to the complexity of the descriptions we are studying. What we propose to do in case of the theory of nescience is to fix a universal reference machine. Since our definition of nescience is a relative measure of what we do not know, intended to compare the topics among themselves, and not an universal measure of unknown, we can fix the an universal Turing machine, and get rid of these constants. The second problem with the Kolmogorov complexity is that it is, in general, a non-computable quantity. As we have said, there are well-defined mathematical concepts that are beyond of the theoretical capabilities of computers. Kolmogorov complexity is one of such mathematical concepts: given a description, there is no algorithm capable of computing the shortest computer program that prints that description. In practice, what we have to do is to approximate the Kolmogorov complexity by the length of the compressed text of the best known description.

**TODO:** How nescience and weak nescience relates to each other

## 13.2 Joint Nescience

We are also interested in our current understanding of the concatenation of two topics.

**Definition 13.2.1** 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 13.2** 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  $s$ . ■

In the theory of nescience it is also needed to know the joint nescience of two topics. Intuitively, the joint nescience of two topics given by our current understanding of both topics, and the shortest string that allows us to effectively reconstruct them.

**Definition 13.2.2 — Joint Nescience.** Let  $t, s \in T$  be two different topics. We define the *joint nescience* of topic  $t$  and  $s$ , denoted by  $v(t, s)$ , as:

$$v(t, s) = (\varepsilon(t, s)^2 + \rho(t, s)^2)^{\frac{1}{2}}$$

Next proposition shows that the combined nescience of two topics is greater or equal than the nescience of any of them isolated.

**Proposition 13.2.1** Given any two topics  $t, s \in T$ , we have that  $v(t, s) \geq v(t)$  and  $v(t, s) \geq v(s)$ .

*Proof.* TODO ■

## 13.3 Conditional Nescience

TODO: somewhere we have to prove this result

**Proposition 13.3.1** We have that  $\sigma(d_t) > \sigma(d_{t|s^*})$  for all topics  $t, s \in \mathcal{T}$  and all  $d_t \in \mathcal{D}_t$ .

*Proof.* TODO ■

TODO: Introduce this concept

TODO: Define the conditional nescience given a descriptio

**Definition 13.3.1 — Conditional Nescience.** Let  $t, s \in T$  be two different topics. We define the *conditional nescience* of topic  $t$  given a perfect knowledge of topic  $s$ , denoted by  $v(t | s)$ , as:

$$v(t | s) = (\varepsilon(t | s)^2 + \rho(t | s)^2)^{\frac{1}{2}}$$

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:** Do the same for the conditional nescience of a description.

**Proposition 13.3.2** We have that  $v(t) > v(t | s)$  for all topics  $t, s \in T$ .

*Proof.* Review:

$$N_t = \frac{l(\hat{d}_t) - K(t)}{K(t)} > \frac{l(\hat{d}_t) - K(t)}{K(t) + l(d_s^*)} = \frac{l(\hat{d}_t) + l(d_s^*) - K(t) - l(d_s^*)}{K(t) + l(d_s^*)} = \frac{l(\langle d_s^*, \hat{d}_t \rangle) - K(t | s)}{K(t | s)} = N_{t|s}$$

■

**TODO:** Prove something like  $N(t, s) = N(t) + N(s|t)$

**TODO:** Check the following proposition

**Proposition 13.3.3**  $N_{t|s} = N_{s|t}$  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 13.3.4** 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 13.3.2** 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*

## 13.4 Nescience of Areas

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 13.4.1** An area  $A$  is a subset of topics, that is  $A \subset T$ .

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 13.3** 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 13.4.2** 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 13.4.3** 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?

## 13.5 Perfect Knowledge

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 13.5.1 — 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 13.4** 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 13.5.1** 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 13.5** In order to clarify how the above theorem can be applied in practice, in Figure 13.1 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. ■

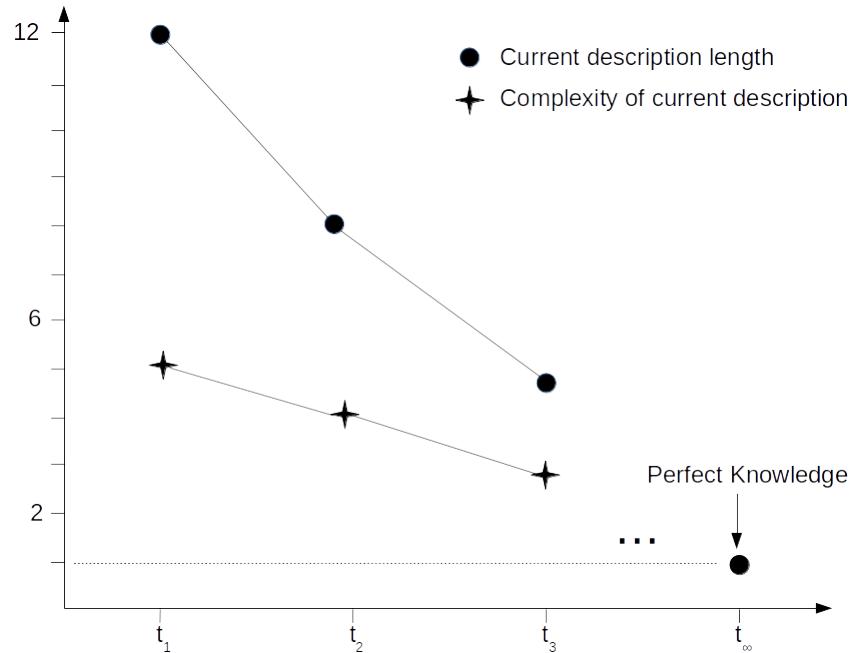


Figure 13.1: In Pursuit of Perfect Knowledge

**TODO:** Introduce the following definition.

**Definition 13.5.2** Let  $s_1, \dots, s_n \in T$  a collection of topics. The *joint nescience* of 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

## 13.6 Current Best Description

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 13.6.1** 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.

### 13.7 Nescience based on Datasets

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 13.7.1** Let  $t \in T$  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 13.7.2** 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 Huffman encoding.

### 13.8 Unknown Unknown

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



## 14. Interesting Questions

*It is not the answer that enlightens,  
but the question.*  
Eugène Ionesco

In this Chapter we propose a set of metrics for the classification of research topics according to their potential as a source of interesting problems, and a methodology for the assisted discovery of new questions. The methodology could be used to identify new applications of existing tools to open problems, and to discover new, previously unconsidered, research topics. The methodology can be applied to both, intradisciplinary and interdisciplinary topics, but the most interesting results arise in case of interdisciplinary topics. In Chapter ?? we will show how the methodology can be applied in practice and we will suggest some new questions and new research topics.

My main assumption is that a research question is interesting if it satisfy the following three requirements:

- A1** The question is new and original (nobody thought about it before).
- A2** Upon its resolution our knowledge about one or more particular research topics increase significantly.
- A3** The practical application of the research results have a large (hopefully positive) impact on people's life.

Some researchers could argue if these are the most convenient requirements. For example, researchers from the so called hard sciences (mathematics, physics, ...) could complain because practical applications are not a critical element in order to pursue an interesting open problem. In those situations we could change the intuitive meaning of the introduced metrics, since they are just mathematical abstractions, and then apply the same methods. The methodology I present 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. In fact, 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

to discover new, previously unseen, objects. The exact definition of concepts like relevance graph, applicability graph or maturity will depend on the area in which the approach is being applied. However, as it was the case of Chapter 13, I prefer to introduce the methodology and the new concepts in the particular context of scientific research, because in this way it is easier to understand them.

As a final note, it is worth to mention the relation of this new methodology to computational creativity and artificial intelligence. What I proposed in this Chapter is an algebraic approach to the *assisted* discovery of potentially interesting questions. I do not intend that the computer will understand the meaning of the posed questions. Moreover, I do not intend that all the questions are relevant or even meaningful. It is up to the researchers to evaluate the proposed combinations of topics and to decide what is exactly the question they suggest, if any.

We have already seen two dimensions for the classification of topics: redundancy (Chapter 12) and error (Chapter 11). This two metrics allow us to quantitatively measure how well we understand a topic, what we call nescience. According to point **A2** of our list of requirements for questions, the higher is the nescience of a topic the higher will be its value as source for interesting research questions.

In this section we are going to introduce two metrics more to characterize topics: relevance and applicability. Relevance measures the impact that a topic has in people's life. Relevance will be used as a complement of nescience. Applicability measures how many times a topic has been used (applied) in other topics. Applicability will allow us to identify new interesting questions.

#### 14.0.1 Combination of Topics

In this section we propose a theoretical model for the process of discovering new research topics by means of the (creative) combination of other, already known, topics. The proposed model is a highly simplified version of what the researchers do in practice. However, it allows us to study some of the properties of this process, and to automate it in practice.

**Definition 14.0.1** Let  $r$  and  $s$  be two topics. A *creative combination* of topics  $r$  and  $s$  is a topic  $t \in \mathcal{T} \setminus \{r, s, \lambda\}$

Let  $r$  and  $s$  be two topics, and let  $CM$  the set of Turing machines that given as input  $r$  and  $s$  produce a new topic  $t \in \mathcal{T}$ . In order to be a *creative combination*, we require that the topic  $t$  must be different from  $r$ ,  $s$  and  $\lambda$ , and the complexity of  $K(t)$  must be greater than the length of the machine. Formally:

$$CM = \{T \in TM : T(r, s) = t, t \in \mathcal{T} \setminus \{r, s, \lambda\}, K(t) \geq l(T)\}$$

Let  $C_{\nabla, f}$  the creative set generated by  $r$  and  $s$ .

**Definition 14.0.2**

**Definition 14.0.3** Let  $r$  and  $s$  be two topics, and let  $T$  be the set of creative Turing machines given  $s$  and  $t$ . We define the creative combination of the topics  $r$  and  $s$ , represented by  $r \oplus s$  is defined as the topic  $t$  produced by the shortest Turing machine, that is:

$$r \oplus s = \min_{l(T)} \{t \in \mathcal{T} : t = T(r, s)\}$$

ddd

**Proposition 14.0.1** The set of topics  $\mathcal{T}$  with the operation of combination  $\oplus$  form a commutative monoid. That is:

- For every two topics  $s$  and  $t$  we have that  $s \oplus t = t \oplus s$ .

- For every three topics  $r, s$  and  $t$  we have that  $r \oplus (s \oplus t) = (r \oplus s) \oplus t$ .

*Proof.* ■

## 14.1 Relevance

Before to measure the impact of a research topic 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 14.1.1** We define the *relevance graph*, denoted by  $RG$ , as the bipartite graph  $RG = (\mathcal{T}, \mathcal{P}, E)$ , where  $\mathcal{T}$  is the set of topics,  $\mathcal{P}$  the set of people, and  $E \subseteq \{(i, j) : i \in \mathcal{T}, j \in \mathcal{P}\}$ . An arc  $(i, j)$  belong to  $E$  if, and only if, person  $j$  is affected by topic  $i$ .

By the set of people  $\mathcal{P}$ , we mean all persons in the World. An edge in the relevance graph means that somebody is affected by a topic, not that this person is interested in that particular topic. The exact meaning of "*being affected by*" is a difficult epistemological concept, and so, it is out the scope of this book. In this sense, our definition of relevance graph is a mathematical abstraction. In Section 18.2 we will see how to approximate in practice this quantity in case of scientific research topics. Alternative interpretations of the relevance graph will be provided in Chapter ?? in the context of the discovery of new business ideas, and Chapter 17 in the area of software engineering.

**■ Example 14.1** A man affected by ALS (amyotrophic lateral sclerosis) disease will be connected in the graph to the ALS topic. The wife of this man will be connected too, because probably she will be suffering the consequences of the disease as well. A researcher interested in ALS will not be connected to ALS node, since he is interested in the disease as a research problem, not suffering its consequences. ■

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 14.1 it is depicted an example of relevance graph.

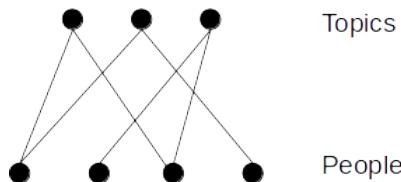


Figure 14.1: Relevance Graph

**Definition 14.1.2** 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,  $R(t) = \deg(t)$ .

**TODO: We should provide a normalized definition of relevance.**

Intuitively, the higher the relevance of a topic, the higher its potential as a source of interesting questions, since we will be working on a problem that affects many people.

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 14.4.

**Proposition 14.1.1** 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 relevance and nescience.

**Definition 14.1.3** Given a topic  $t \in \mathcal{T}$ , we define the *interestingness of the topic as a problem*, denoted by  $IP(t)$ , as:

$$IP(t) = v(t)R(t)$$

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.

**TODO:** Explain why we prefer the product of both quantities, instead of the Euclidean distance to the origin.

■ **Example 14.2** 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. ■

## 14.2 Applicability

As we mention in Example 14.2, the fixed point theorem is not very interesting as a research problem by itself. However, it is a very important theorem, since it can be applied to prove many other theorems. In this section we are going to introduce the concept of *applicability*, a new measure that allows us to identify which topics are important because they can be used as tools.

In order to formally define the concept of applicability, first we have to introduce a new graph that describes which topics have been applied as tools in other topics.

**Definition 14.2.1** 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  $(i, j)$  belongs to  $E$  if, and only if, topic  $t_i$  has been applied to topic  $t_j$ .

For example, there is 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.

**TODO:** Say something about how the applicability graph can be computed in practice.

**TODO:** Include a picture of applicability graph.

**Definition 14.2.2** We define the *applicability* of a topic  $t \in T$ , denoted by  $A(t)$ , as the outdegree of that node in the applicability graph, that is:

$$A(t) = \text{outdeg}(t)$$

Intuitively, the higher the applicability of a topic, the higher is its potential as a tool that can be applied to solve new problems. If a tool has been already applied many times in the past to solve open problems, it is likely that it can be applied to solve other open problems as well.

**TODO:** We should provide a normalized definition of applicability.

**TODO:** Mention here we can use other, alternative, centrality measures to compute applicability.

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 14.2.1** 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 14.1.1. ■

to provide a new metric to measure how *mature* is a topic.

**Definition 14.2.3** Given a topic  $t \in \mathcal{T}$ , we define the *maturity* of topic  $t$ , denoted as  $M(t)$ , as the inverse of nescience, that is:

$$M(t) = v(t)^{-1}$$

Intuitively, the more mature is a topic the higher is its potential applicability to solve other open problems. Highly immature topics should not be applied to solve open problems, since they could provide wrong solutions.

**TODO:** We should provide a normalized definition of relevance.

■ **Example 14.3** **TODO:** Find an illuminating example. ■

Finally, we can 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.

**Definition 14.2.4** Let  $t \in \mathcal{T}$  a topic. We define the *interestingness of the topic as a tool*, denoted by  $IT(t)$ , as:

$$IT(t) = A(t)M(t)$$

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.

**TODO:** Again, mention why the product.

■ **Example 14.4** **TODO** ■

## 14.3 Interesting Questions

The set of topics  $\mathcal{T}$  is composed by all possible topics, including those topics we are not yet aware of its existence. However, in order to pose interesting questions, we have to restrict our search to only those topics we already know. How to discover new interesting research topics will be described in Section XX.

**Definition 14.3.1** The set of *known research topics*, denoted by  $T'$ , is the subset of  $T$  composed by all those topics that have been the subject of a research activity.

A question is a pair of topics  $t_1, t_2 \in T'$ . Intuitively, the question would be something like "can we apply the tool described by topic  $t_1$  to solve the problem described by topic  $t_2$ ?".

**Definition 14.3.2** Given two topics  $t_1, t_2 \in T'$ , a *question*, denoted by  $Q_{t_1 \rightarrow t_2}$ , is the ordered pair  $(t_1, t_2)$ .

The most interesting questions will arise when topic  $t_1$  has a high interestingness as a tool, and a topic  $t_2$  has a high interestingness as a problem. The interestingness of the new question is defined as a function of the interestingness of the topics involved.

**Definition 14.3.3** The *interestingness* of the question  $Q_{t_1 \rightarrow t_2}$ , denoted by  $IQ_{t_1 \rightarrow t_2}$ , is given by:

$$IQ_{t_1 \rightarrow t_2} = A_{t_1} R_{t_2} + M_{t_1} N_{t_2}$$

In practice, what we have to do is to compute all the possible combination of those topics with very large *IT* with those other topics with very large *IP*, and select those combinations with higher *IQ*. Of course, most of the questions posed with this approach will be meaningless, but the same happens when researchers organize brainstorming sessions to identify new tools to solve difficult problems.

We can easily generalize the above procedure to multiple tools and, perhaps, multiple problems. In this way we came up with the application of two tools to a given problem ( $T + T \rightarrow P$ ), the application of a single tool to the combination of two problems ( $T \rightarrow P + P$ ), and so on. The exact meaning of those combinations of tools and problems depends on the topics themselves, and so, it is left to the creative interpretation of the researcher.

From a practical point of view, sometimes it is convenient to restrict ourselves to certain areas to find interesting questions, or even to specific topics. For example, if a researcher is specialist in area  $A$ , he might be interested in finding interesting problems where he can apply his knowledge. In the same way, a researcher specialist in problem  $t$  could be interested in finding new tools to solve the problem.

**Definition 14.3.4** Let  $A \subset T'$  a research area, and  $t_1, t_2 \in T'$  two topics. If both topics belong to the same area, that is  $t_1, t_2 \in A$ , we say that the question  $Q_{t_1 \rightarrow t_2}$  is *intradisciplinary*, otherwise, we say that the question is *interdisciplinary*.

In principle, the most innovative questions would be the 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. Interdisciplinary questions would address my requirement **A1** of what makes a question interesting, that is, the question is new and original.

## 14.4 New Research Topics

In the previous section my focus was in how to find new interesting research questions. In this section I will go one step beyond, and I will show how to identify new, previously unconsidered, research topics.

**Definition 14.4.1** 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 14.4.2** Lets  $T'$  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 14.4.1** 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 14.4.1 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 14.1.1), a possible approach to identify new topics could be by means of combining already existing interesting problems. In Figure 14.2 is depicted graphically the idea.

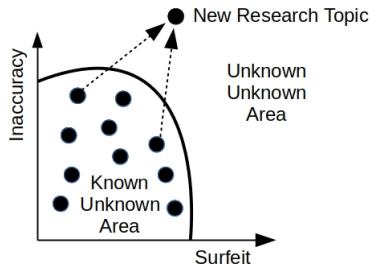


Figure 14.2: The discovery of new research topics

**Definition 14.4.3** 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 14.4.4** 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  $IP_t$  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 14.4.5** 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 thought about them is lower.

## 14.5 Classification of Research Areas

In the same way we studied the nescience of research areas (see Section 13.4), we could also study the interestingness of research areas.

**Definition 14.5.1** 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 humankind<sup>1</sup> and, in general, not very well understood.

■ **Example 14.5** 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 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 13 and 14).

<sup>1</sup>Please 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.

**References**

Popper and falsificationism





## 15. 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.

### 15.1 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.

### 15.1.1 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 A). 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 15.1.1** 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 15.1.2** 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 15.1.3 — 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 15.1.4 — 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 numbers<sup>1</sup>. 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

---

<sup>1</sup>TODO: 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 15.1.** *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 15.2.** *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 15.1.1** 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.

### 15.1.2 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 A.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 16.1 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 C), 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`<sup>2</sup>, 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

---

<sup>2</sup>Do not confuse the  $\lambda$  symbol with a  $\lambda$ -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  $\lambda$ . 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  $\lambda$ -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  $\lambda$  symbol. That  $\lambda$  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  $\lambda$ . In order to prove that  $\lambda$  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  $\lambda$  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.

### 15.1.3 Category Theory

TODO: Interpretation of the theory of nescience in terms of category theory.

## 15.2 Scientific Method

Next definition formally introduces the concept of scientific methodology in the context of the theory of nescience.

**Definition 15.2.1** 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.)

### 15.2.1 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 15.2.1** 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 15.2.2** 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 15.2.2** 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 15.2.3** 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  $\delta_a$  and  $\delta_b$  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 15.2.4** 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 15.2.5** 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 15.2.6** 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  $\delta$ . Study the invariants under  $\delta$ .

## 15.3 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.

## 15.4 Science vs. Pseudoscience

TODO: Provide a characterization of the difference between science and pseudoscience. In science, nescience decreases with time, in pseudoscience not.

## 15.5 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 15.5.1 — 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 15.5.1** 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 5.3.1. ■

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 15.5.2** 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 15.5.3** Let  $A \subset T$  a research area. The *graspness* of area  $A$ , denoted by  $G(A)$ , is defined by:

$$G(A) = \frac{1}{d(A)} \sum_{t \in A} G(t)$$

## 15.6 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.

## 15.7 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 15.7.1** 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.

## 15.8 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

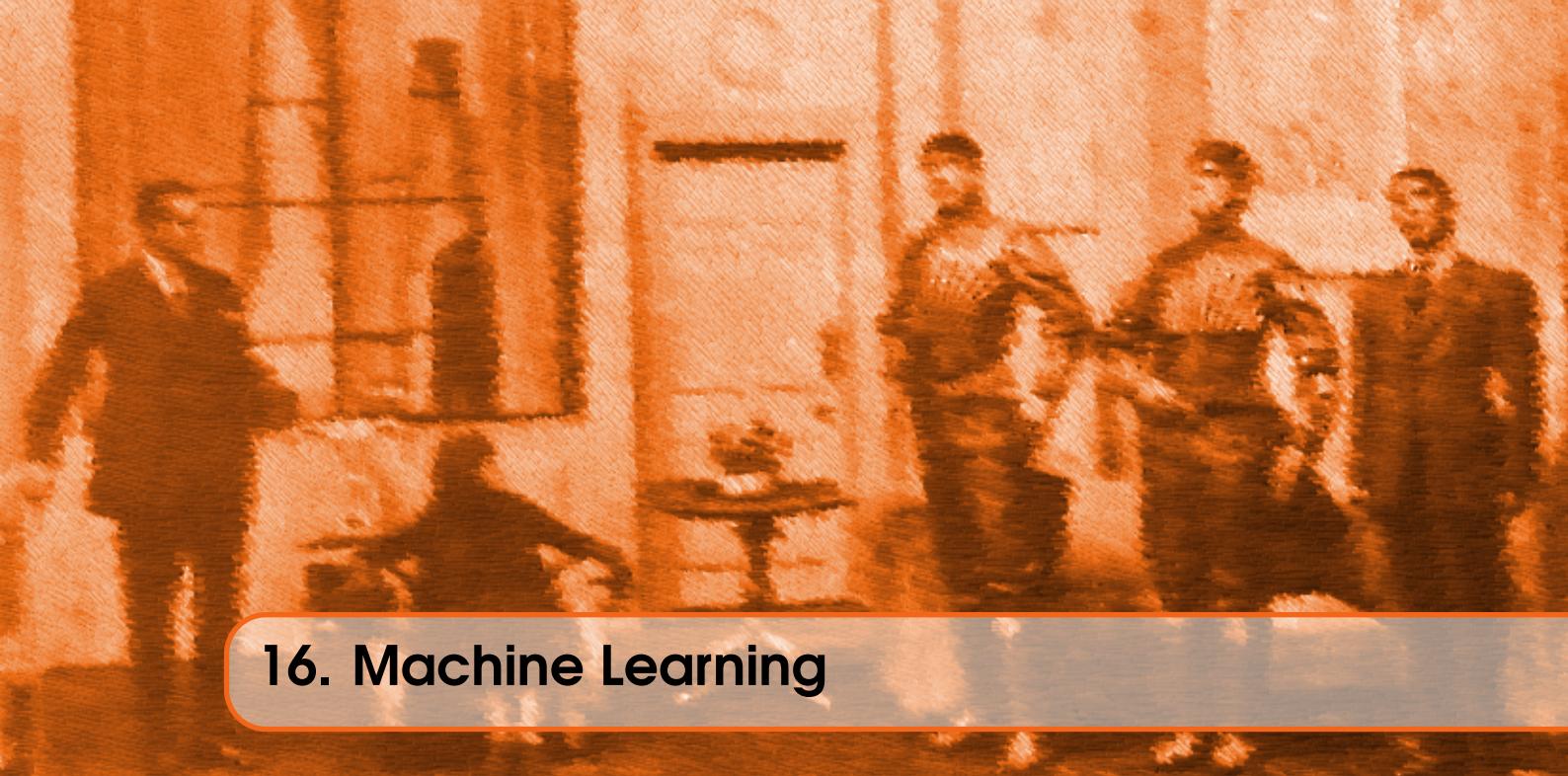
Mention the polemic between Hilbert and Fregé given a reference to the book of Mosterin.



# Part 3: Applications

<b>16</b>	<b>Machine Learning .....</b>	<b>195</b>
16.1	Nescience Python Library	
16.2	A Note About Compression	
16.3	Miscoding	
16.4	Inaccuracy	
16.5	Surfeit	
16.6	Nescience	
16.7	Auto Machine Classification	
16.8	Auto Machine Regression	
16.9	Time Series	
16.10	Anomaly Detection	
16.11	Decision Trees	
16.12	Algebraic Model Selection	
16.13	The Analysis of the Incompressible	
<b>17</b>	<b>Software Engineering .....</b>	<b>235</b>
17.1	Redundancy of Software	
17.2	Quality Assurance	
17.3	Forex Trading Robots	
<b>18</b>	<b>Philosophy of Science .....</b>	<b>243</b>
18.1	Wikipedia, The Free Encyclopedia	
18.2	Classification of Research Topics	
18.3	Classification of Research Areas	
18.4	Interesting Research Questions	
18.5	Interesting Research Topics	
18.6	Philosophy of Science	
18.7	Evolution of Knowledge	
18.8	Graspness of Topics	
18.9	Probability of Being True	
18.10	Unused text	
<b>19</b>	<b>Computational Creativity .....</b>	<b>259</b>
19.1	Classification of Research Topics	
19.2	Interesting Research Questions	
19.3	New Research Topics	
19.4	Classification of Research Areas	
19.5	References	
19.6	Future Work	





## 16. 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`<sup>1</sup> library, an open source python library that provides an implementation of the ideas included in this chapter.

### 16.1 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

<sup>1</sup><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.

## 16.2 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 5.4). 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 16.2.1** 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 16.1**

■

(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.

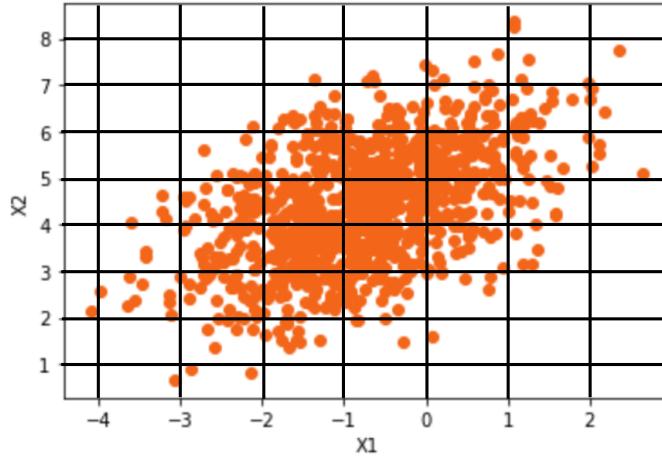


Figure 16.1: Uniform discretization.

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.

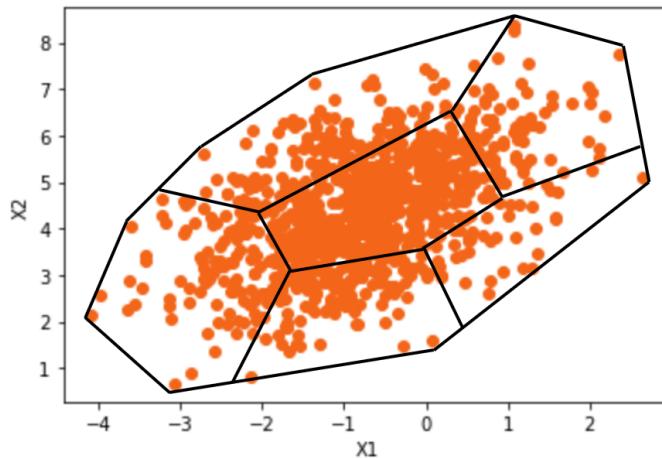


Figure 16.2: Convex hull and kmeans discretization.

### 16.3 Miscoding

In Section 10.1.3 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 6.4):

$$E(\mathbf{x}_j, \mathbf{y}) = \frac{\max\{K(\mathbf{x}_j | \mathbf{y}), K(\mathbf{y} | \mathbf{x}_j)\}}{\max\{K(\mathbf{x}_j), K(\mathbf{y})\}}$$

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 16.2).

**Definition 16.3.1** 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 16.2** 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 following<sup>2</sup>:

```
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 16.3** 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 16.3 we can see a two-dimensional projection of this dataset, along the hyperplane composed by features 8 and 10.

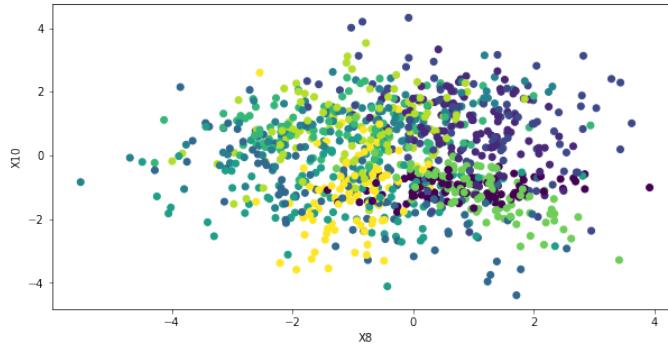


Figure 16.3: Gaussian Blob Cluster.

```
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 16.4) 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.

<sup>2</sup>Since we are generating a list of 1.000 random samples, the reader could get a slightly different result when running this example.

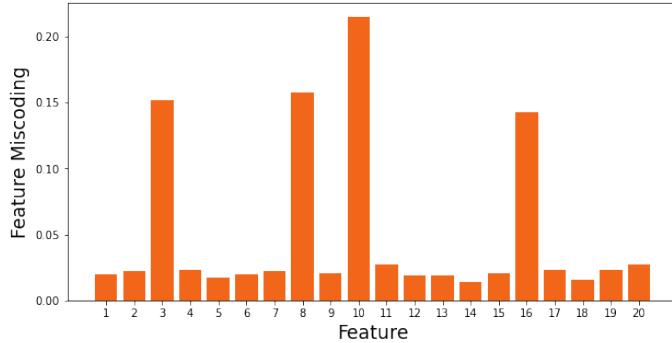


Figure 16.4: Miscoding of a Synthetic Dataset.

We can compare miscoding with correlation, a common technique used in machine learning to identify the most relevant features of a dataset. In Figure 16.5 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 ( $\mathbf{x}_3$ ).

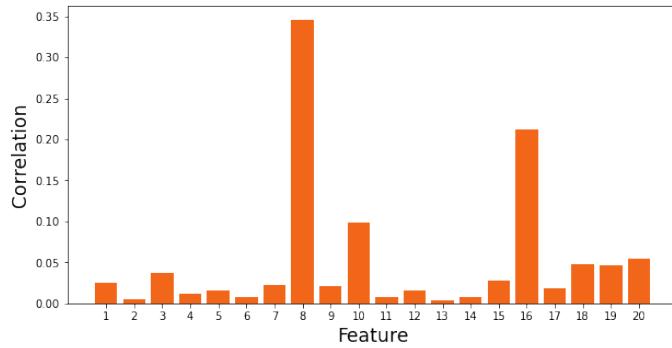


Figure 16.5: Correlation of a Synthetic Dataset.

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 16.3.2** 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 16.4** We will use again the synthetic dataset of Example 16.3, 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 16.6, 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.

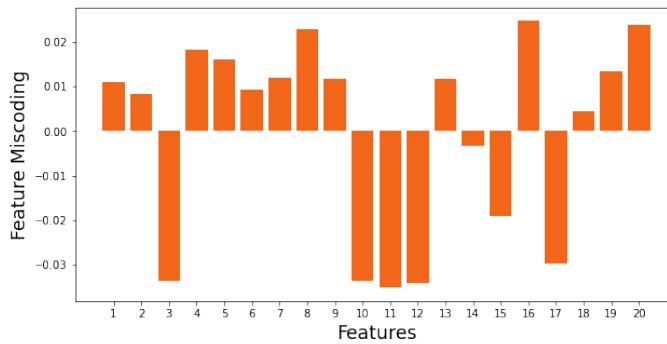


Figure 16.6: Partial Feature Miscoding.

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 16.3.3** 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 16.5** Based on the dataset and the partial features miscoding computed in Example 16.4, in Figure 16.7 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 16.6** 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

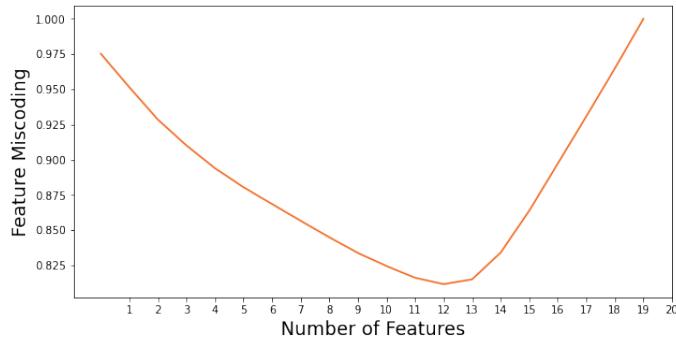


Figure 16.7: Accumulated Partial Feature Miscoding.

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 16.5 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 16.7).

**■ Example 16.7** 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 16.8 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.

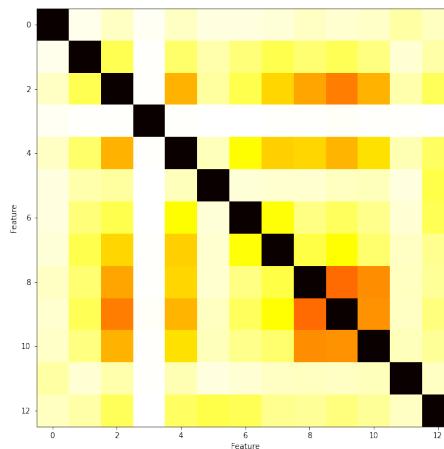


Figure 16.8: Regular Miscoding Matrix.

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.

## 16.4 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 16.4.1** 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 16.8** 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 16.9, both metrics present a similar behavior, having inaccuracy a larger value, due to a stronger emphasis in incorrectly predicted values.

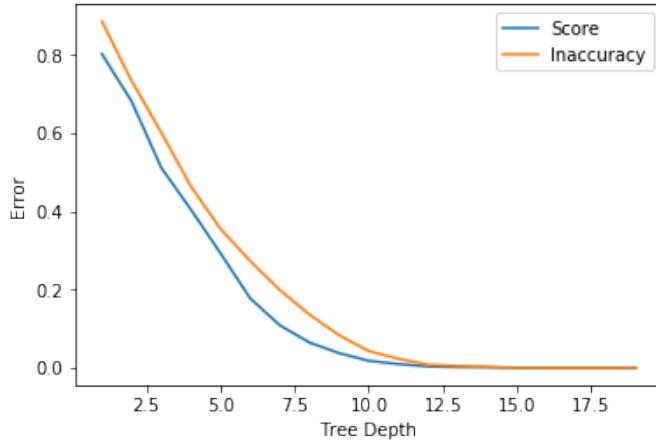


Figure 16.9: Inaccuracy vs. Score of Decision Trees

In Example 16.8 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 16.9).

■ **Example 16.9** 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 increased<sup>3</sup> from 0.17 to 0.21.

<sup>3</sup>Note 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 16.9, 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 16.10 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 16.10** 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 16.10 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.

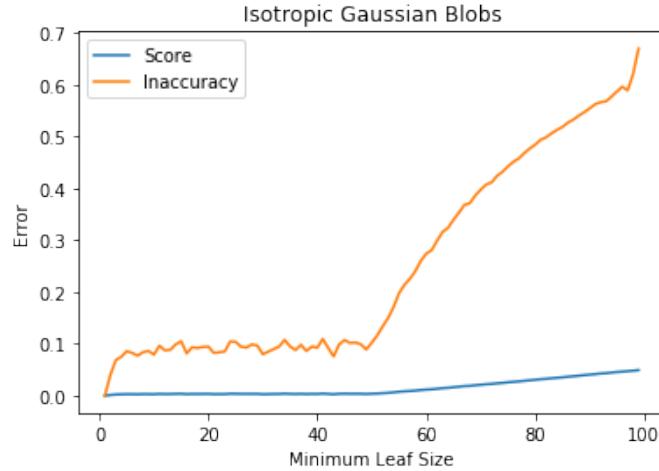


Figure 16.10: Inaccuracy of Decision Tree.

■

## 16.5 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 16.5.1** 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 16.11 shows.

■ **Example 16.11** 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 16.11 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 16.12** In this example we are going to compare a decision tree with a neural network.

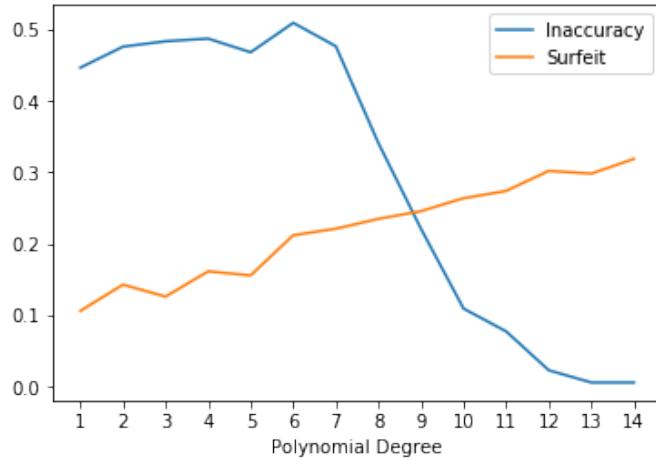


Figure 16.11: Surfeit vs Inaccuracy

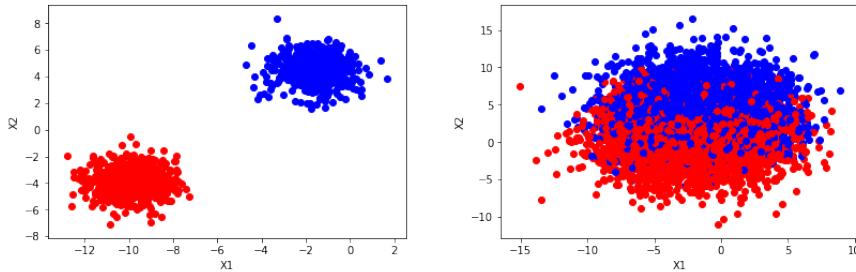


Table 16.1: Isotropic Gaussian blobs.

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 16.1, 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 16.1, 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 16.12 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.

## 16.6 Nescience

In Chapter 13 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 16.13 shows how we can use the nescience library to compute the nescience of a dataset and a model.

■ **Example 16.13** 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 16.13 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 16.14.

■ **Example 16.14** 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 16.15).

■ **Example 16.15** 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 16.12. 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.

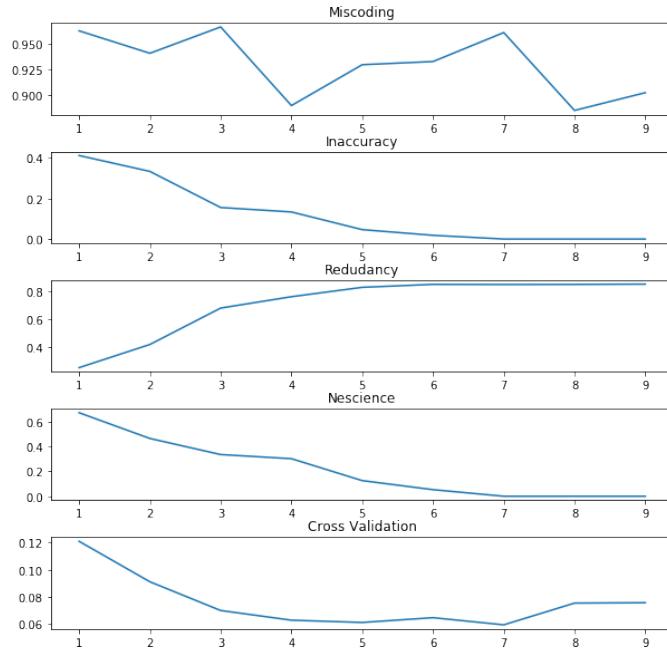


Figure 16.12: Evolution of Nescience with Tree Depth.

It is interesting to note the behavior of the three metrics that define the concept of nescience in Figure 16.12. 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 16.11.

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 16.16 it is show how we can do that with the nescience library.

■ **Example 16.16** 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.

■

## 16.7 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 16.17** 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.

### 16.7.1 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,<sup>4</sup> `[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.

## 16.8 Auto Machine Regression

### 16.9 Time Series

■ TODO: Introduce this section

#### 16.9.1 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 16.9.1** 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 16.18** In this example we are going to study the presence of cycles in the number of passengers of a US airline. In Figure 16.13 is depicted a time series of monthly passengers from 1949 to 1960 (AirPassenger dataset, see References section bellow). As we can observe, there is

---

<sup>4</sup>If 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.

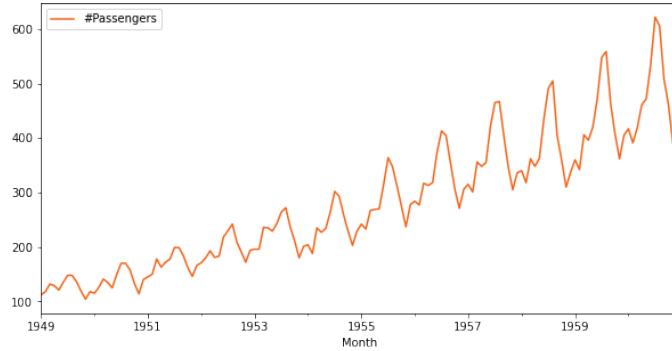


Figure 16.13: Air Passengers.

```
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 16.14 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).

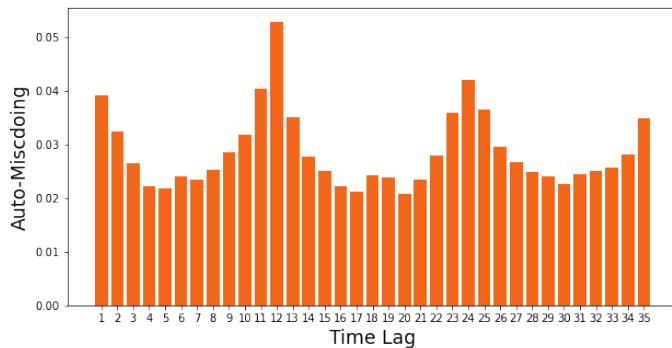


Figure 16.14: Auto-miscoding of Air Passengers.

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 16.9.2** 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 16.19** 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 16.15 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.

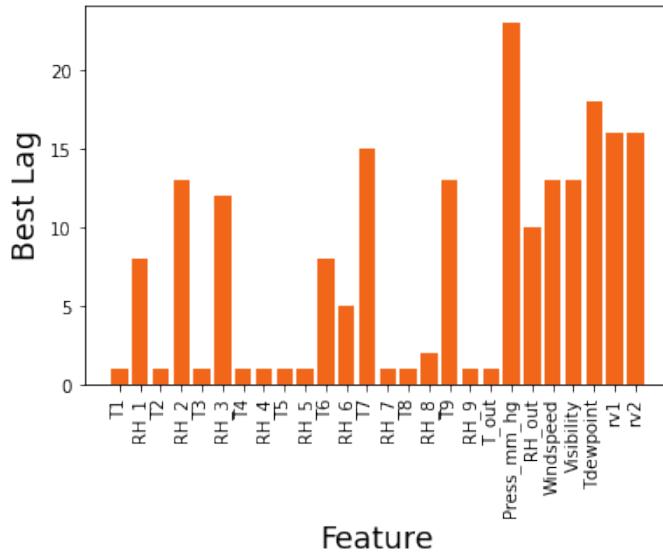


Figure 16.15: Cross MisCoding Lag

**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.

### 16.9.2 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 16.9.3** 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

$$\mathbf{y}_t = \mathbf{Z}_t \boldsymbol{\alpha}_t + \mathbf{d}_t + \boldsymbol{\varepsilon}_t \quad \boldsymbol{\varepsilon}_t \sim N(0, \mathbf{H}_t) \quad (16.1)$$

called *space? observation or measurement equation*, and

$$\boldsymbol{\alpha}_t = \mathbf{T}_t \boldsymbol{\alpha}_{t-1} + \mathbf{c}_t + \mathbf{R}_t \boldsymbol{\eta}_t \quad \boldsymbol{\eta}_t \sim N(0, \mathbf{Q}_t) \quad (16.2)$$

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  $\alpha_t$  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  $\varepsilon_t$  and  $\eta_t$  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  $\alpha_1$  is assumed to be generated as  $\alpha_1 \sim NID(a_1, P_1)$ , independen of the observation and estate disturbances  $\varepsilon_t$  and  $\eta_t$ . 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  $\alpha_t$  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 16.20**

**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 16.9.4**

$$\begin{aligned} \mathbf{a}_{t+1} &= \mathbf{T}_t \mathbf{a}_t + \mathbf{K}_t \mathbf{v}_t \\ \mathbf{K}_t &= \mathbf{T}_t \mathbf{P}_t \mathbf{Z}_t^T \mathbf{F}_t^1 \\ \mathbf{v}_t &= \mathbf{y}_t - \mathbf{Z}_t \mathbf{a}_t \end{aligned} \tag{16.3}$$

called *prediction equations*, and

$$\begin{aligned} \mathbf{F}_t &= \mathbf{Z}_t \mathbf{P}_t \mathbf{Z}_t^T + \mathbf{H}_t \\ \mathbf{L}_t &= \mathbf{T}_t - \mathbf{K}_t \mathbf{Z}_t \\ \mathbf{P}_{t+1} &= \mathbf{T}_t \mathbf{P}_t \mathbf{L}_t^T + \mathbf{R}_t \mathbf{Q}_t \mathbf{R}_t^T \end{aligned} \tag{16.4}$$

called *updating equations*.

■ **Example 16.21** **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 16.22** **TODO:** Provide an example of auto-time series ■

## 16.10 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 16.10.1** 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 16.23 we will see how to apply this class to indentify houses with abnormal low prices, and to explain why they are cheaper.

■ **Example 16.23** 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")
```

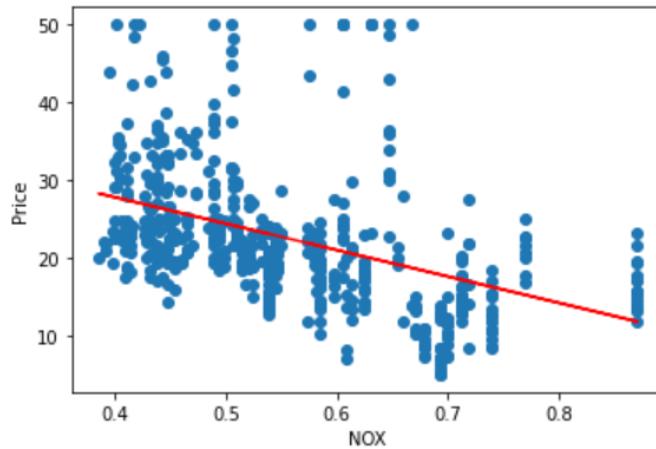


Figure 16.16: Price as a function of NOX.

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.

## 16.11 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

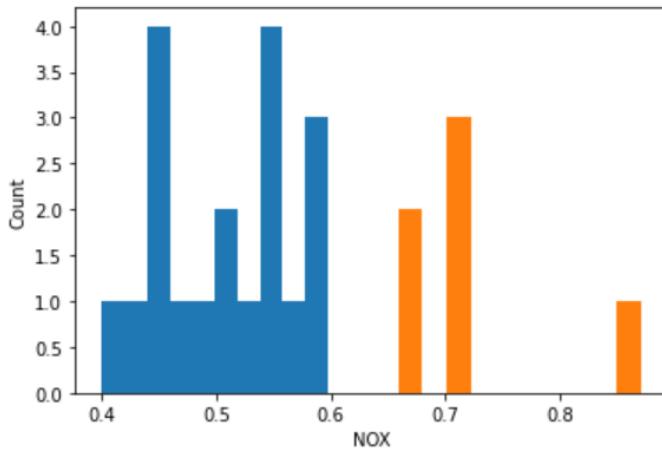


Figure 16.17: Histogram of anomalies NOX.

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.

### 16.11.1 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 16.6), 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 16.3 and Section 16.4 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 16.7.1. 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 16.5), 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  $\mathbf{Q}_l = \{\mathbf{x}_i \in \mathbf{Q} : x_{ij} = w\}$ <sup>5</sup>.

<sup>5</sup>Ideally, 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

$$\begin{aligned}\tilde{H}(\mathbf{Q}, \theta) &= \frac{n_l}{n} \left( - \sum_{i=1}^k \frac{\sum_{j=1}^{n_l} I(y_j^l = g_i)}{n_l} \log_2 \frac{\sum_{j=1}^{n_l} I(y_j^l = g_i)}{n_l} \right) \\ &\quad + \frac{n_r}{n} \left( - \sum_{i=1}^k \frac{\sum_{j=1}^{n_r} I(y_j^r = g_i)}{n_r} \log_2 \frac{\sum_{j=1}^{n_r} I(y_j^r = g_i)}{n_r} \right)\end{aligned}\quad (16.5)$$

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 book<sup>6</sup> 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)
```

<sup>6</sup><http://www.mathematicsunknown.com>

```
print("Score: ", model.score(data.data, data.target))
```

### 16.11.2 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 16.18 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.

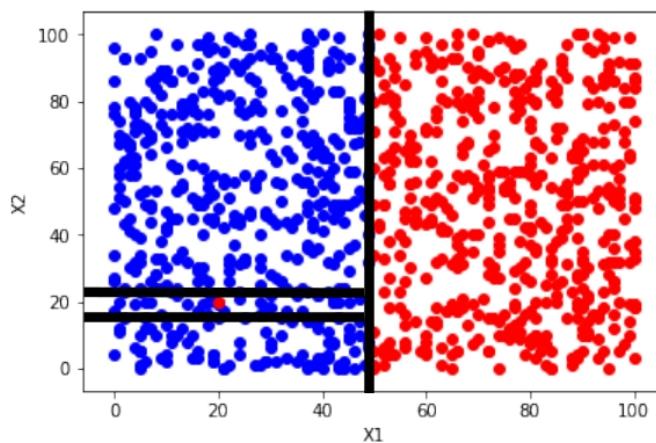


Figure 16.18: Synthetic dataset with CART algorithm splits.

The tree obtained by applying our algorithm to the dataset of Figure 16.18 can be seen in Figure 16.19. 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.

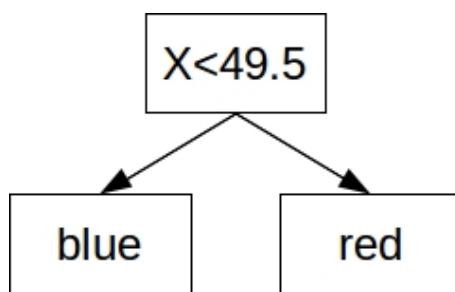


Figure 16.19: Decision tree obtained by the nescience algorithm.

Our second experiment, again with synthetic dataset, is depicted in Figure 16.20. 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 16.21.

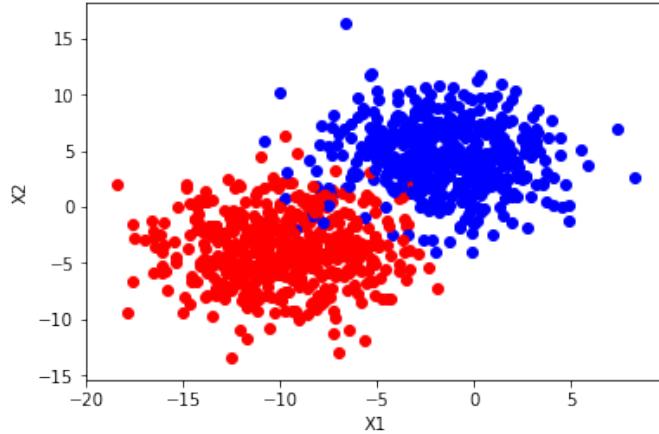


Figure 16.20: Isotropic Gaussian Blobs.

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).

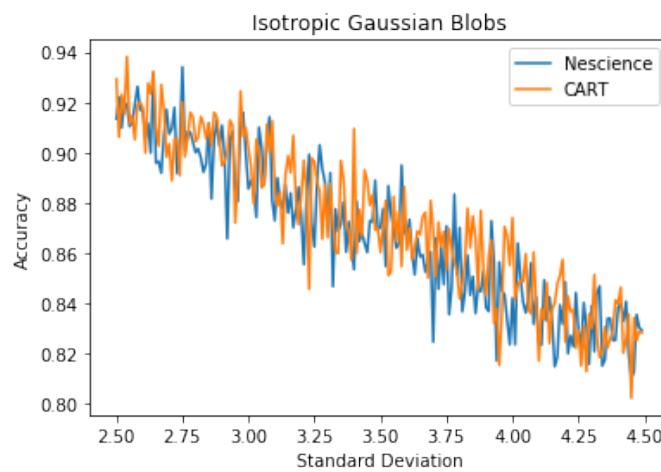


Figure 16.21: Accuracy of Isotropic Gaussian Blobs.

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 16.22. 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).

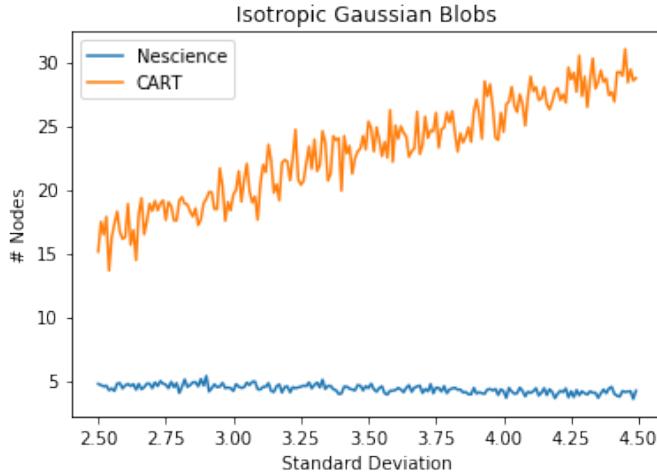


Figure 16.22: Number of Nodes.

In Figure 16.23 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.

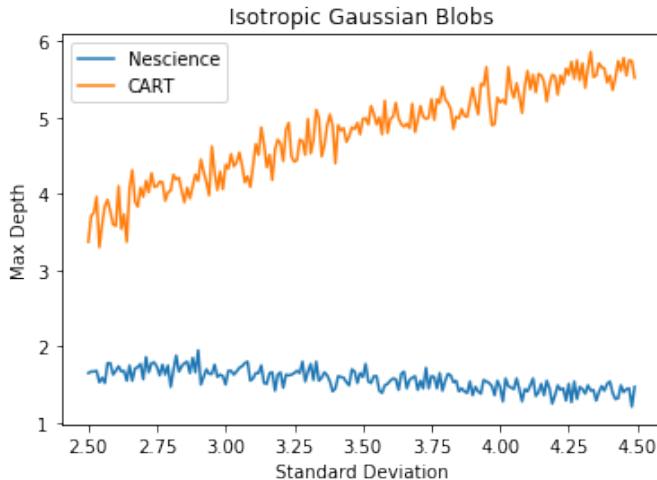


Figure 16.23: Maximum depth of the model.

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.

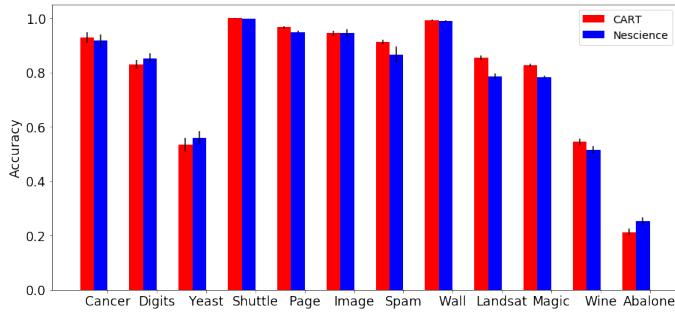


Figure 16.24: Maximum depth of the model.

In Figure 16.24 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.

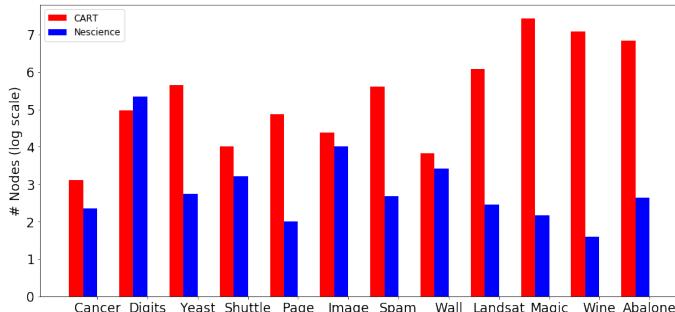


Figure 16.25: Maximum depth of the model.

In Figure 16.25 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).

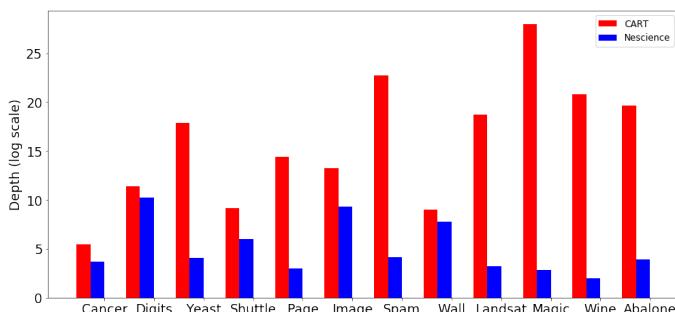


Figure 16.26: Maximum depth of the model.

Finally, in Figure 16.26 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 16.11.2, 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.

	<b>Euclid</b>	<b>Arithmetic</b>	<b>Geometric</b>	<b>Product</b>	<b>Addition</b>	<b>Harmonic</b>
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 16.2: Comparison of nescience functions

Similarly, Table 16.11.2 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.

	<b>bz2</b>	<b>lzma</b>	<b>zlib</b>
Accuracy	0.813	0.804	0.81
Stdev	0.03	0.045	0.038

Table 16.3: 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).

## 16.12 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 16.27).

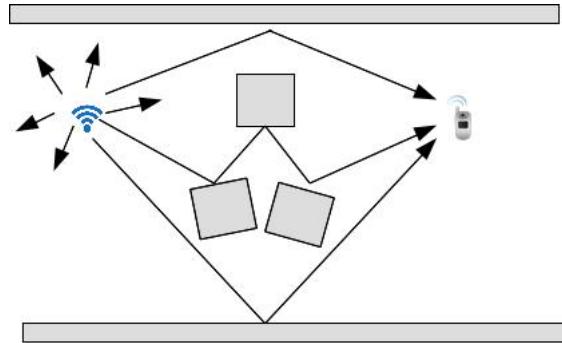


Figure 16.27: Multipath Signal Propagation

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  $\sigma$  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 16.28) 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 Kismet<sup>7</sup> 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.

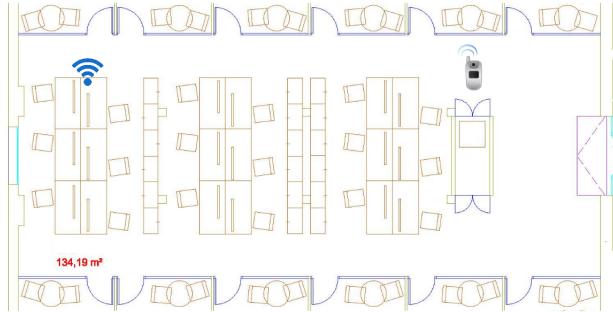


Figure 16.28: Experimental Set Up

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 16.4: 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.

<sup>7</sup><https://www.kismetwireless.net/index.shtml>

## 16.13 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  $\underline{\mathcal{X}}$ , 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 [Shm+10].

The application of the minimum description length principle to the identification of optimal decision trees have been proposed in [QR89], further refined and clarified in [WP93]; 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 [ ].

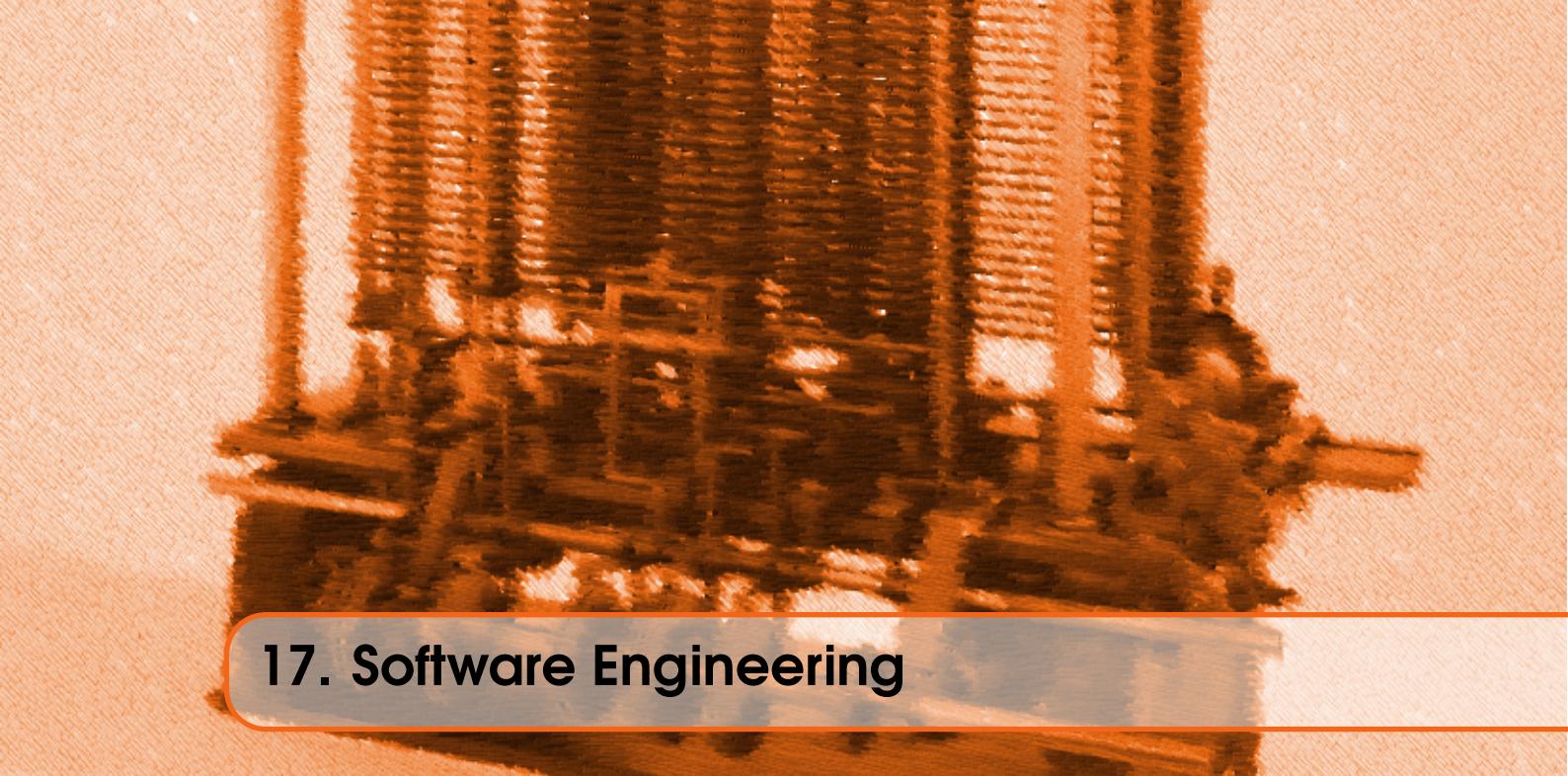
The Minimum Description Length [Grü07] and the Minimum Message Length [Wal05] techniques have been applied to the problem of inferring decision trees in [QR89], later on clarified and extended in [WP93], in [MRA+95] as a technique for pruning, and in [RS98], among others. Although the underlining concepts 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 [Gar+13]; 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 [YW09], 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>



## 17. 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 fist 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 task 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 project, 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.

### 17.1 Redundancy of Software

The concept of nescience applied to software engineering allow 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 exists 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**

SQLite<sup>1</sup>, 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 17.1 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 '.*.[chyl]' -exec cat {} \; > total-1.0
```

Figure 17.1 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**

PostgreSQL<sup>2</sup> 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.

---

<sup>1</sup><http://www.sqlite.org>

<sup>2</sup><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 17.1: Nescience of SQLite

Table 17.2 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 17.1. The command issued to compute the number of source lines was:

```
# find postgresql-1_0/src -name '.*.[chyl]' -exec cat {} \; > total-1.0
```

Figure 17.2 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 Cassandra<sup>3</sup> 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.

---

<sup>3</sup><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 17.2: Nescience of PostgreSQL



Figure 17.1: Nescience of SQLite

Table 17.3 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 17.1 and Table 17.2, 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 17.3 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.

## 17.2 Quality Assurance

Table 17.4 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 17.5 shows the ten functions with higher nescience. **TODO: explain**

Finally, Table 17.6 shows the ten most interesting functions from the point of view of testing. **TODO: explain**

## 17.3 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 17.3: 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 17.4: 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 17.5: Highest nescience in SQLite functions

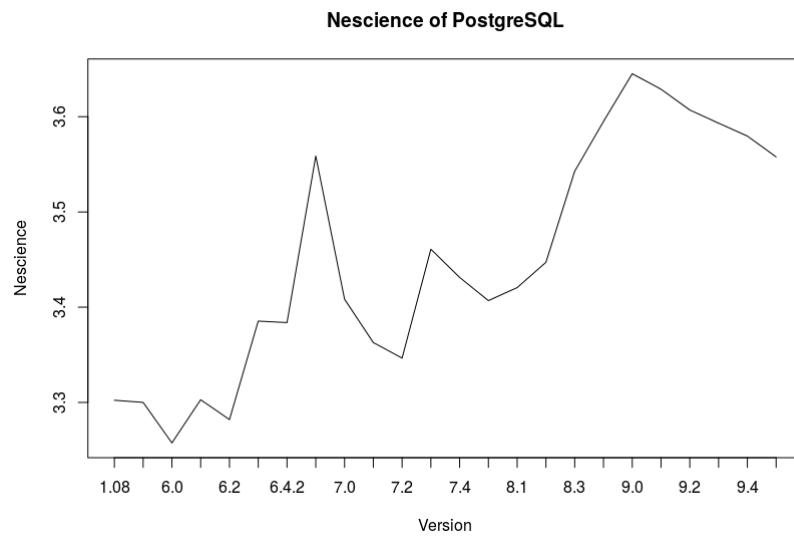


Figure 17.2: Nescience of PostgreSQL

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 17.6: Interestingness of SQLite functions

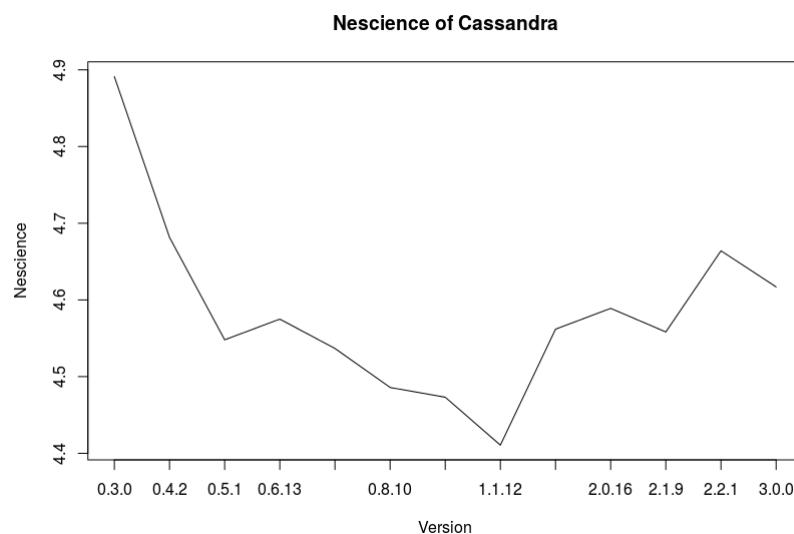
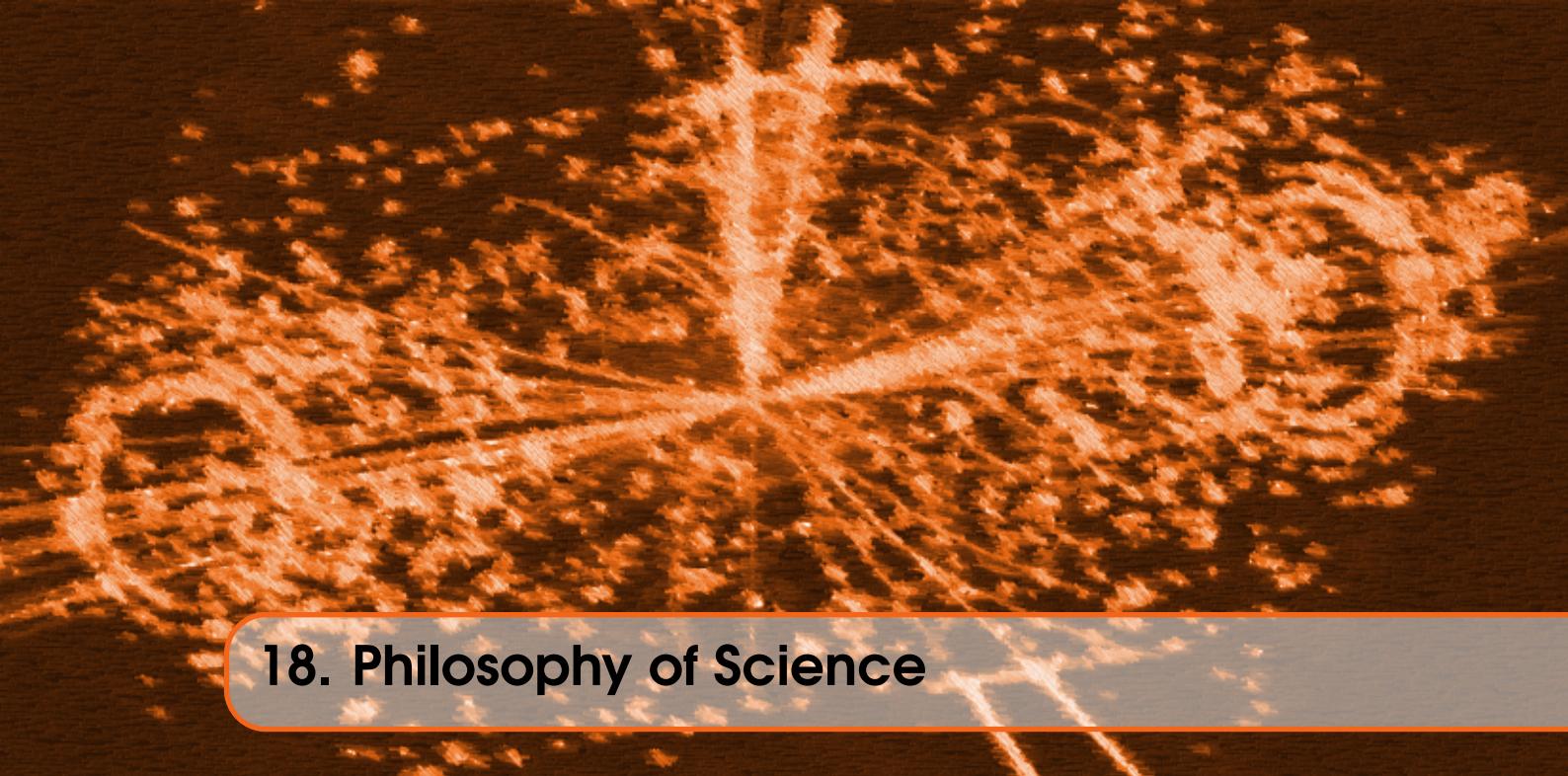


Figure 17.3: Nescience of Cassandra





## 18. Philosophy of Science

*Science may be regarded as the art of data compression.*  
Li & Vitányi

### 18.1 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.

### 18.2 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 18.1 shows a plot of nescience for all the topics after normalization.

Table 18.1 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

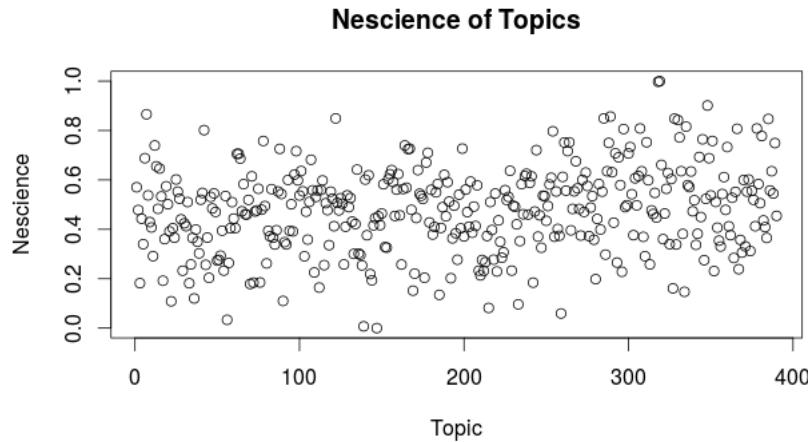


Figure 18.1: Nescience of topics

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 18.1: 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 an 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 18.2 shows a plot of the maturity of the selected set of topics after the normalization process.

Table 18.2 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*

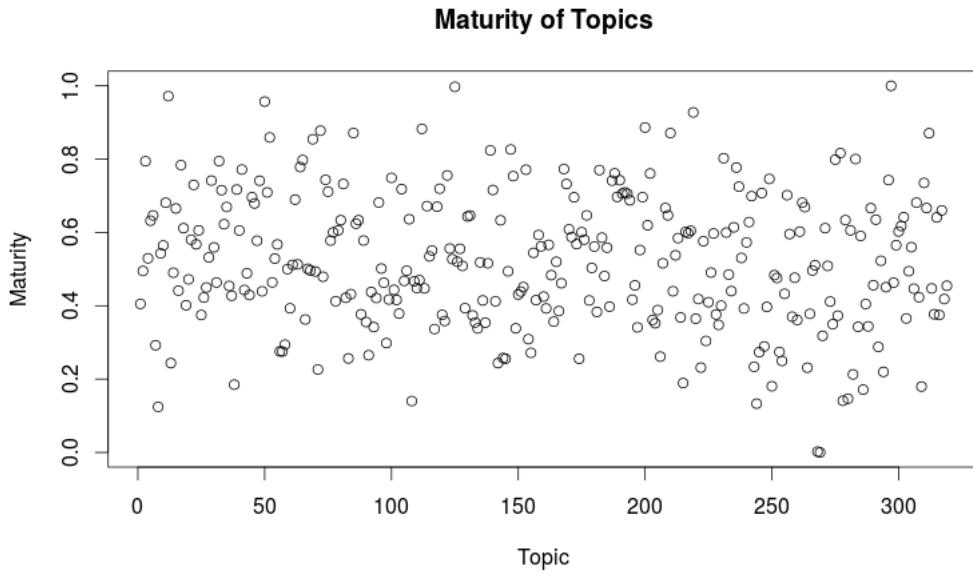


Figure 18.2: Maturity of topics

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 18.2: 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 18.3 shows a plot of the interestingness of the selected set of topics after the normalization process.

Table 18.3 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

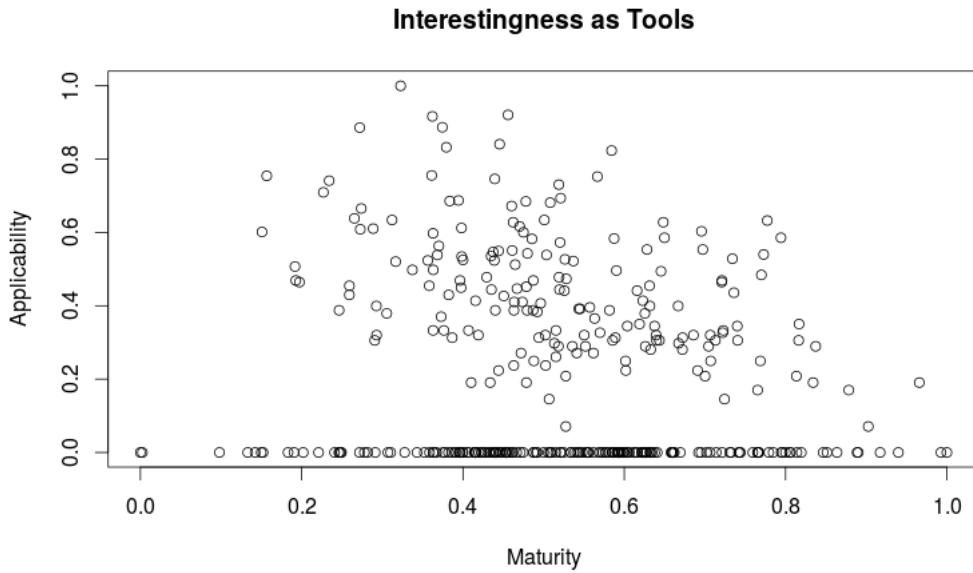


Figure 18.3: Interestingness of Tools

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 18.3: 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 18.4 contains a plot of the interestingness of the topics considered as potential interesting problems.

Table 18.4 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

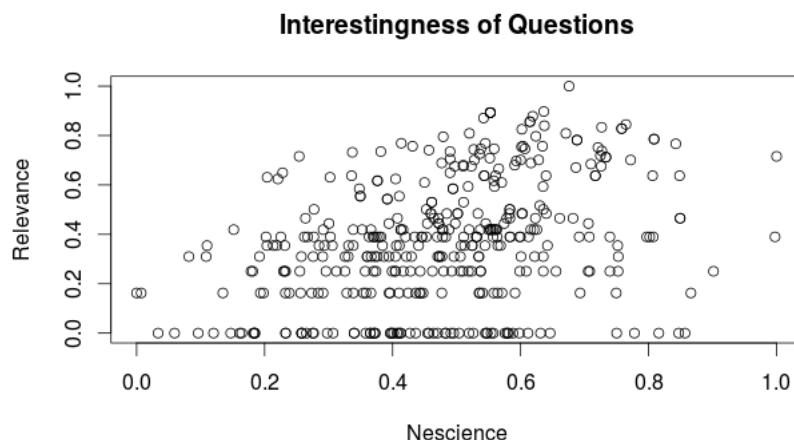


Figure 18.4: Interestingness of Questions

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 18.4: 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.

### 18.3 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 18.5 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

Research Area	Applicability	Maturity	Tools
Sociology	$1.00 \times 10^{-3}$	$2.93 \times 10^{-3}$	$3.09 \times 10^{-3}$
Biology	$9.20 \times 10^{-4}$	$4.65 \times 10^{-3}$	$4.74 \times 10^{-3}$
Chemistry	$3.11 \times 10^{-3}$	$5.01 \times 10^{-3}$	$5.90 \times 10^{-3}$
Psychology	$1.14 \times 10^{-3}$	$6.91 \times 10^{-3}$	$7.00 \times 10^{-3}$
Mathematics	$9.32 \times 10^{-3}$	$9.47 \times 10^{-3}$	$1.32 \times 10^{-2}$
Epistemology	$1.55 \times 10^{-3}$	$1.75 \times 10^{-2}$	$1.76 \times 10^{-2}$
Computer_science	$9.93 \times 10^{-3}$	$1.90 \times 10^{-2}$	$2.15 \times 10^{-2}$

Table 18.5: Interestingness of Areas as Tools

	Relevance	Nescience	Problems
Mathematics	$4.22 \times 10^{-2}$	$3.51 \times 10^{-1}$	$3.53 \times 10^{-1}$
Computer_science	$2.35 \times 10^{-2}$	$4.43 \times 10^{-1}$	$4.44 \times 10^{-1}$
Chemistry	$5.95 \times 10^{-2}$	$4.66 \times 10^{-1}$	$4.70 \times 10^{-1}$
Biology	$3.85 \times 10^{-2}$	$4.75 \times 10^{-1}$	$4.77 \times 10^{-1}$
Psychology	$5.06 \times 10^{-2}$	$5.28 \times 10^{-1}$	$5.31 \times 10^{-1}$
Epistemology	$4.54 \times 10^{-2}$	$5.30 \times 10^{-1}$	$5.32 \times 10^{-1}$
Sociology	$4.21 \times 10^{-2}$	$5.43 \times 10^{-1}$	$5.44 \times 10^{-1}$

Table 18.6: Interestingness of Areas as Problems

area of highest interest, and sociology is the area with less interest. The only strange elements 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 18.6 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.

## 18.4 Interesting Research Questions

By combining the elements of Table 18.3 and Table 18.4 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 18.7.

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

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 18.7: Interesting Intradisciplinary Questions

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 18.8: Interesting Interdisciplinary Questions

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 18.8 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 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.

## 18.5 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 18.5 it is shown a plot<sup>1</sup> of the interestingness of the (potential) new topics compared with the interestingness of the topics that generated them.

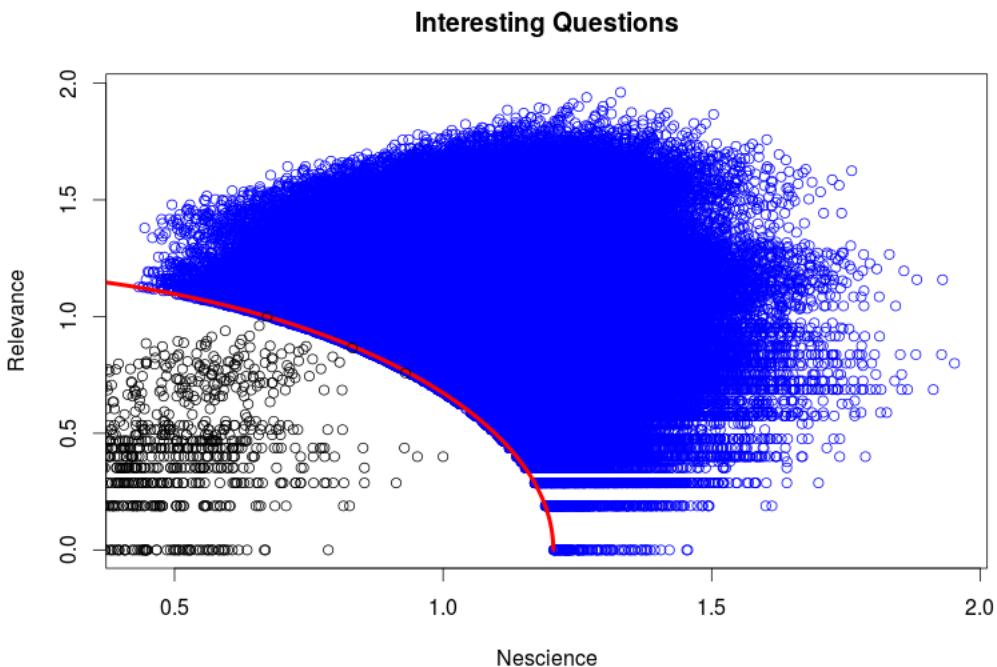


Figure 18.5: Interesting Intradisciplinary Questions

Table 18.9 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 18.10 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*

<sup>1</sup>With 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 18.9: New Topics

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 18.10: Restricted New Topics

*machines* could be defined as the union of a group of Turing machines that are linked together 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.

## 18.6 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:

- 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 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 18.7, 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.

## 18.7 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 18.1** In Figure 18.6 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. ■

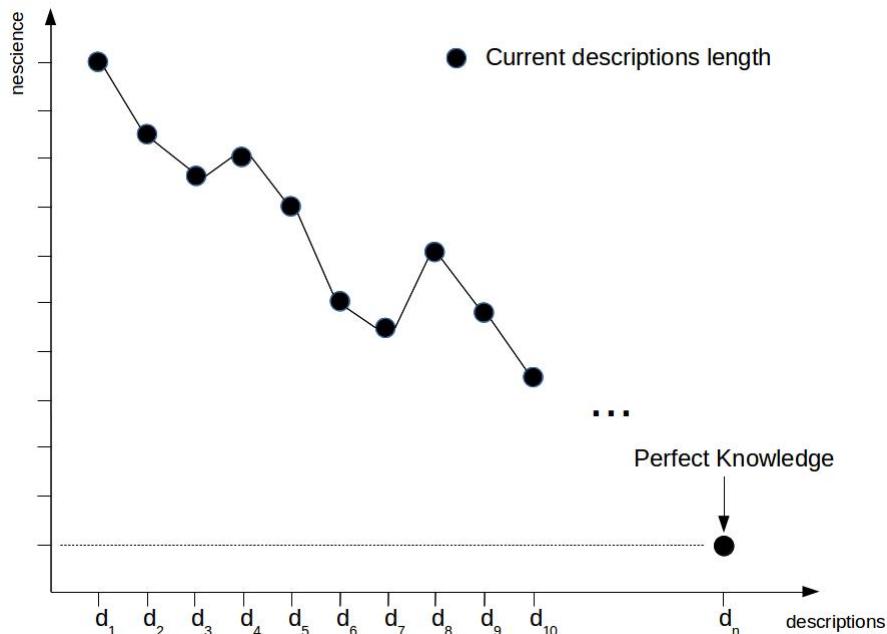


Figure 18.6: Evolution of Nescience

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 *sicence 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 18.2** Suppose that the topic  $t$  being studied is the concept of *limit of a function*. The standard  $(\varepsilon, \delta)$ -*definition* of limit provided by Karl Weierstrass is:

$$\lim_{x \rightarrow c} f(x) = L \Leftrightarrow \forall \varepsilon > 0 \exists \delta > 0 : \forall x (0 < |x - c| < \delta \Rightarrow |f(x) - L| < \varepsilon)$$

This definition has a length of 46 symbols, spaces not included<sup>2</sup>. 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 hyperreal<sup>3</sup> 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.  $\square$  ■

## 18.8 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 an easy to understand topic

## 18.9 Probability of Being True

**TODO:** Study the probability of being true.

**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 18.4** **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?

---

<sup>2</sup>

**■ Example 18.3** 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. ■

<sup>3</sup>Nonstandard 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.

## 18.10 Unused text

Here I should mention the work of Solomonoff with respect to assign a prior probability to theories. 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 an 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 [CAO+05] with applications to the normalized compression distance (a measure of similarity between objects proposed in [Li+04]).





## 19. 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 14 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.

## 19.1 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  $\mu_t$  refers to the considered metric (nescience, relevance, ...).

### 19.1.1 Relevance

In Definition 14.1.2 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 14.1.1). 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 19.1 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.

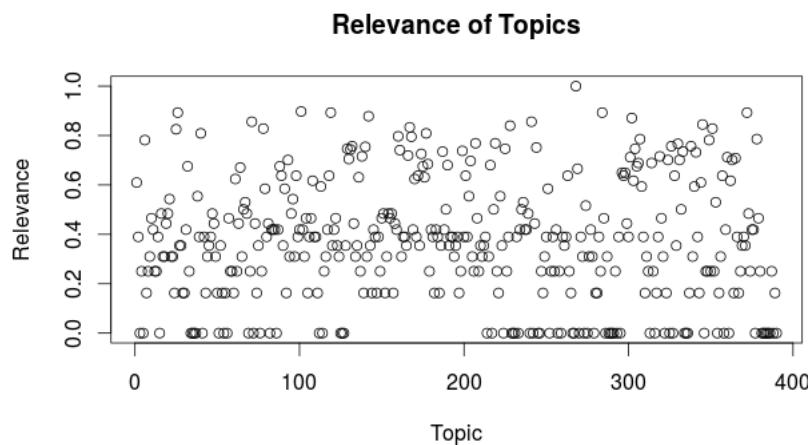


Figure 19.1: Relevance of topics

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 19.1: Relevance of topics

Table 19.1 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*).

### 19.1.2 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 14.2.1), and applicability is formally defined as the out-degree of the topic in this graph (see Definition 14.2.2). 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 19.2 shows a plot of the applicability of the selected set of topics after the normalization process, and Table 19.2 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.

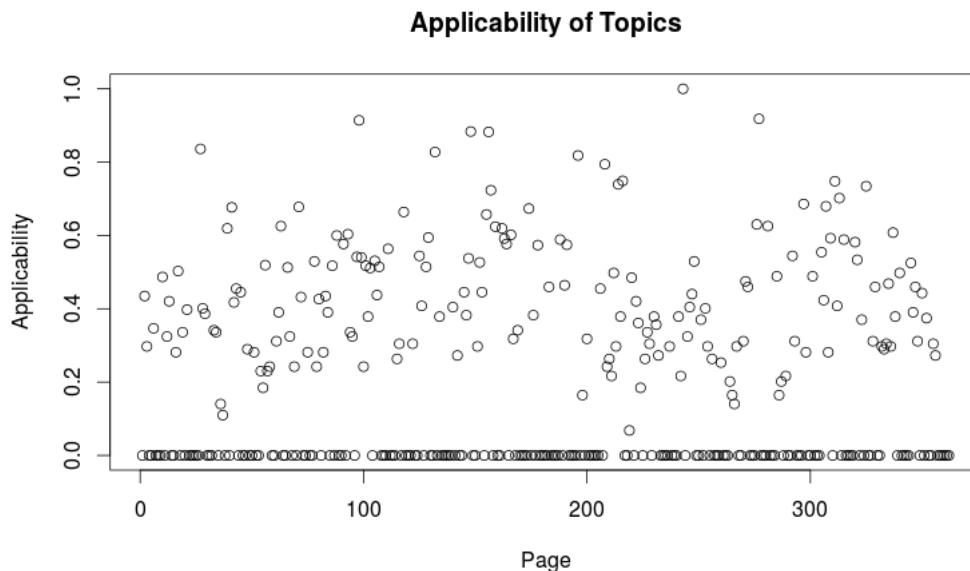


Figure 19.2: Applicability of topics

### 19.1.3 Maturity

## 19.2 Interesting Research Questions

### 19.2.1 Intradisciplinary Questions

### 19.2.2 Interdisciplinary Questions

## 19.3 New Research Topics

## 19.4 Classification of Research Areas

## 19.5 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 19.2: Applicability of topics

## 19.6 Future Work



# Appendix

<b>A</b>	<b>Foundations of Mathematics .....</b>	<b>267</b>
A.1	Propositional Logic	
A.2	Predicate Logic	
A.3	Set Theory	
A.4	Lambda Calculus	
A.5	Category Theory	
<b>B</b>	<b>Advanced Mathematics .....</b>	<b>275</b>
B.1	Distinctiveness of Metrics	
B.2	Distinctiveness of Metrics	
<b>C</b>	<b>Coq Proof Assistant .....</b>	<b>281</b>
<b>D</b>	<b>About Quotes and Photos .....</b>	<b>285</b>
<b>E</b>	<b>Notation .....</b>	<b>291</b>
	<b>Bibliography .....</b>	<b>295</b>
	Books	
	Articles	
	<b>Index .....</b>	<b>299</b>



## A. 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.

### A.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 A.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 A.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 A.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 A.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 A.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 A.1.1. The list  $A_1, \dots, A_n$  is denoted by  $\mathcal{S}$ .

**Definition A.1.4** An *assignment* of  $\mathcal{S}$ , denoted by  $\mathcal{A}(\mathcal{S})$  or simply  $\mathcal{A}$  if there is no doubt about the  $\mathcal{S}$  to which we are referring, is a mapping of the sentences of  $\mathcal{S}$  to a list of  $n$  true values.

An assignment  $\mathcal{A}(\mathcal{S})$  can be extended to all the sentences of  $\mathcal{F}(\mathcal{S})$  by means of computing the corresponding true tables. The true value of a sentence  $F$  given the assignment  $\mathcal{A}(\mathcal{S})$  is denoted by  $\mathcal{A}(F)$ .

**Definition A.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 A.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 A.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 A.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 A.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 A.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 A.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.

## A.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 A.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 A.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 A.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 A.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 A.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 A.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  $\varphi$  is a formula, then  $\neg\varphi$  is a formula.

**R2** If  $\varphi$  and  $\psi$  are formulas then  $\varphi \wedge \psi$  is a formula.

**R3** If  $\varphi$  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 A.2.4** Let  $x$  be a variable and  $\varphi$  a formula. The symbol  $\forall$ , called *for all*, is defined as:

1  $\forall x\varphi$  abbreviates  $\neg\exists x\neg\varphi$ .

#### ■ Example A.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.

## A.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.

<sup>1</sup>It called axiom schema because there exists one axiom for each  $P$ .

**Axiom of Choice** Every family of nonempty sets has a choice function.

## A.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  $\lambda$  called *abstractor*, and the parenthesis  $( , )$ . The list  $v_1, v_2, \dots, v_n$  is not a set, in the sense of Section A.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 A.4.1** A  $\lambda$ -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 A.4.1 encapsulates the way that functions are constructed. We have an *application* procedure that given the  $\lambda$ -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  $\Lambda$  the collection of all the  $\lambda$ -terms that can be constructed using Definition A.4.1. By convention, outermost parentheses are not written.

### ■ Example A.5

We use the letters  $x, y, z$  and variants with subscripts and primes to denote variables in  $V$ . To denote elements of  $\Lambda$ , we use  $L, M, N, P, Q, R$  and variants thereof. Syntactically identity of two  $\lambda$ -terms will be denoted by the symbol  $\equiv$ .

**Definition A.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  $\lambda$ -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 A.4.3**  $L$  is a proper subterm of  $M$  if  $L$  is a subterm of  $M$ , but  $L \not\equiv M$ .

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  $\lambda$ .

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  $\beta$ -reduction.

Reduction can be extend 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.

**A.5 Category Theory**



## B. 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.

$$\log_a x = \frac{1}{\log_b a} \log_b x \quad (\text{B.1})$$

TODO: And introduce the following inequation

$$\ln \frac{1}{x} \geq 1 - x \quad (\text{B.2})$$

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.

### B.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  $a$ .

## B.2 Distinctiveness of Metrics

### B.2.1 Accuracy

Accuracy is the ratio between the number of correct predictions and the total number of testing values.

**Definition B.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 B.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 B.2.2** The order of total number of possible values for the metric accuracy is:

$$\mathcal{O}(m)$$

■

*Proof.* Given Proposition [prop:Accuracy].

■

### B.2.2 Precision

Precision is the ratio between the correctly predicted positive values and the total number of predicted positive values.

**Definition B.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 B.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 B.2.4** The order of total number of possible values for the metric precision is:

$$\mathcal{O}(m^2)$$

*Proof.* Given Proposition [prop:Precision]. ■

### B.2.3 Recall

Recall measures the ratio between the number of correctly predicted positive values and the total number of positives.

**Definition B.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 B.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 B.2.6** The order of total number of possible values for the metric recall is:

$$\mathcal{O}(m)$$

*Proof.* Given Proposition [prop:Recall]. ■

### B.2.4 F1

$F_1$  is the harmonic mean between precision and recall.

**Definition B.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 B.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 B.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]. ■

### B.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 B.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 B.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

$$\binom{m}{p} = \frac{m!}{n!p!} = \frac{m(m-1)\cdots(m-p+1)}{p(p-1)\cdots1}.$$

The denominator is the constant  $m!$ . ■

**Corollary B.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 B.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 B.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 B.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.

## C. 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  $O(s) < O(t)$ .



## D. 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

### D.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**

## D.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 GIMP<sup>1</sup>, 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

---

<sup>1</sup>[www.gimp.org](http://www.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.

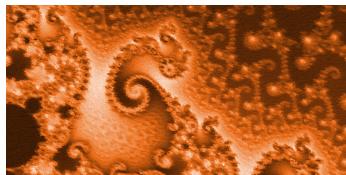
## Morse key



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."

## E. 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

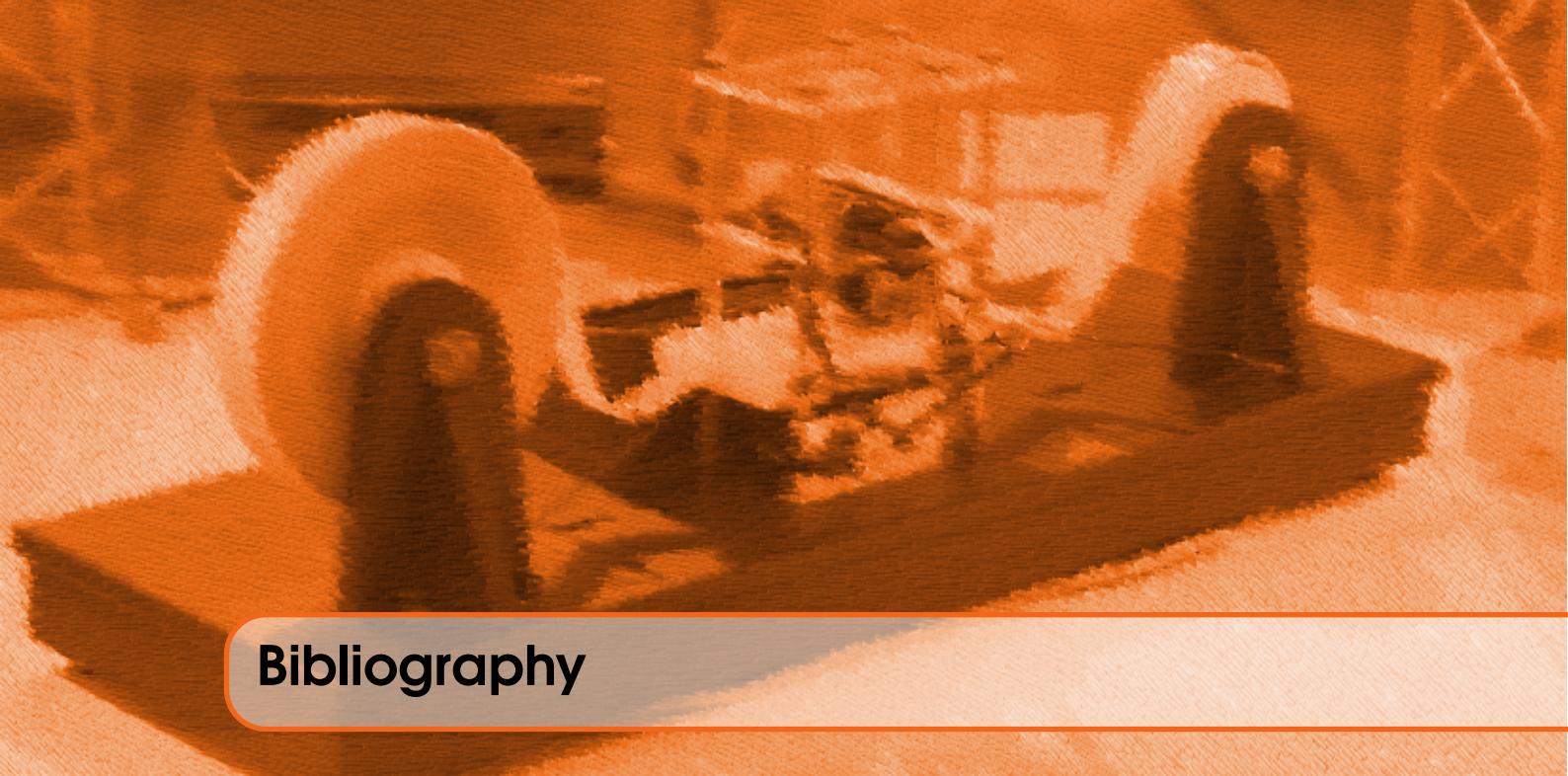
- $\mathbb{N}$  set of natural numbers (including 0)
- $\mathbb{Z}$  set or integers
- $\mathbb{Z}^+$  set or positive integers
- $\mathbb{Q}$  set of rational numbers
- $\mathbb{R}$  set of real numbers
- $\mathbb{R}^+$  set of positive real numbers
- $x \in A$   $x$  is a member of  $A$
- : set formation
- $A \subset B$   $A$  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
$\lambda$	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
$\Omega$	sample space
$P(x)$	probability of $x$
$E(X)$	expectation of $X$
$T$	Turing machine
$Q$	set of states
$\Gamma$	set of tape symbols
$\sqcup$	blank symbol
$\Sigma$	input symbols
$q_o$	initial state
$q_f$	final state
$\tau$	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 $t \in \mathcal{T}$
$\delta$	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

- [Abr63] Norman Abramson. *Information theory and coding*. 1963 (cited on pages 89, 157).
- [BKW05] David A Belsley, Edwin Kuh, and Roy E Welsch. *Regression diagnostics: Identifying influential data and sources of collinearity*. Volume 571. John Wiley & Sons, 2005 (cited on page 37).
- [Coo03] S Barry Cooper. *Computability theory*. CRC Press, 2003 (cited on page 76).
- [CT12] Thomas M Cover and Joy A Thomas. *Elements of information theory*. John Wiley & Sons, 2012 (cited on page 89).
- [DH97] Anthony Christopher Davison and David Victor Hinkley. *Bootstrap methods and their application*. Volume 1. Cambridge university press, 1997 (cited on page 37).
- [DeG+86] Morris H Morris H DeGroot et al. *Probability and statistics*. 04; QA273, D4 1986. 1986 (cited on page 47).
- [Epp10] Susanna S Epp. *Discrete mathematics with applications*. Cengage Learning, 2010 (cited on page 47).
- [Fer09] Maribel Fernández. *Models of Computation: An Introduction to Computability Theory*. Springer Science & Business Media, 2009 (cited on page 76).
- [GG12] Allen Gersho and Robert M Gray. *Vector quantization and signal compression*. Volume 159. Springer Science & Business Media, 2012 (cited on page 89).
- [Grü07] Peter D Grünwald. *The minimum description length principle*. MIT press, 2007 (cited on page 233).
- [Jam+13] Gareth James et al. *An introduction to statistical learning*. Volume 112. Springer, 2013 (cited on page 37).
- [Jec13] Thomas Jech. *Set theory*. Springer Science & Business Media, 2013 (cited on page 139).

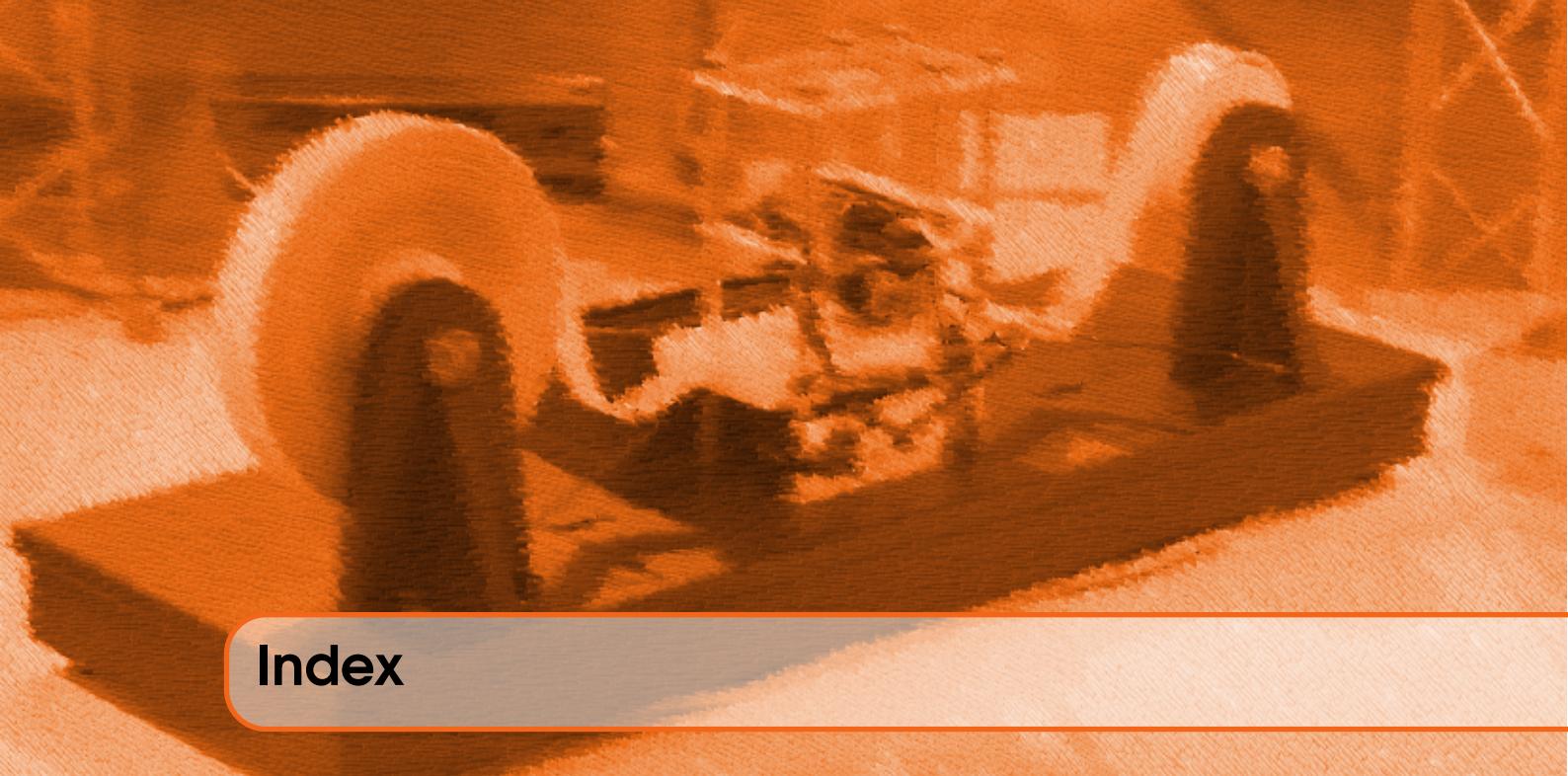
- [Kau13] Perry J Kaufman. *Trading systems and methods*. John Wiley & Sons, 2013 (cited on page 37).
- [LV13] Ming Li and Paul Vitányi. *An introduction to Kolmogorov complexity and its applications*. Springer Science & Business Media, 2013 (cited on page 140).
- [Mos16] Jesús Mosterín. *Conceptos y teorías en la ciencia*. Alianza, 2016 (cited on page 37).
- [Nag14] Jennifer Nagel. *Knowledge: A very short introduction*. OUP Oxford, 2014 (cited on page 37).
- [Par92] Robert Pardo. *Design, testing, and optimization of trading systems*. John Wiley & Sons, 1992 (cited on page 37).
- [RK95] Kenneth H Rosen and Kamala Krithivasan. *Discrete mathematics and its applications*. Volume 6. McGraw-Hill New York, 1995 (cited on page 47).
- [Sip12] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, 2012 (cited on page 76).
- [Soa16] Robert I Soare. *Turing computability*. Springer, 2016 (cited on page 76).
- [Spi+12] Murray R Spiegel et al. *Probability and statistics*. Volume 4. Mcgraw-hill, 2012 (cited on page 47).
- [Wal05] Christopher S Wallace. *Statistical and inductive inference by minimum message length*. Springer Science & Business Media, 2005 (cited on pages 115, 233).

## Articles

- [CJF00] Marie-Christine Cadiergues, Christel Joubert, and Michel Franc. “A comparison of jump performances of the dog flea, *Ctenocephalides canis* (Curtis, 1826) and the cat flea, *Ctenocephalides felis felis* (Bouché, 1835)”. In: *Veterinary parasitology* 92.3 (2000), pages 239–241 (cited on page 37).
- [CAO+05] Manuel Cebrián, Manuel Alfonseca, Alfonso Ortega, et al. “Common pitfalls using the normalized compression distance: What to watch out for in a compressor”. In: *Communications in Information & Systems* 5.4 (2005), pages 367–384 (cited on page 257).
- [Cha69] Gregory J Chaitin. “On the simplicity and speed of programs for computing infinite sets of natural numbers”. In: *Journal of the ACM (JACM)* 16.3 (1969), pages 407–422 (cited on page 98).
- [Cha95] Gregory J Chaitin. “The berry paradox”. In: *Complexity* 1.1 (1995), pages 26–30 (cited on page 140).
- [Cor+15] Héctor Cordobés de la Calle et al. “Empirical comparison of graph-based recommendation engines for an apps ecosystem”. In: *International Journal of Interactive Multimedia and Artificial Intelligence* 3.2 (2015), pages 33–39 (cited on page 37).
- [Cre82] Noel Cressie. “Playing safe with misweighted means”. In: *Journal of the American Statistical Association* 77.380 (1982), pages 754–759 (cited on page 37).
- [Gar+13] Salvador Garcia et al. “A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 25.4 (2013), pages 734–750 (cited on page 233).
- [Göd31] Kurt Gödel. “Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I”. In: *Monatshefte für mathematik und physik* 38.1 (1931), pages 173–198 (cited on page 139).

- 
- [HR78] David Harrison and Daniel L Rubinfeld. “Hedonic housing prices and the demand for clean air”. In: *Journal of environmental economics and management* 5.1 (1978), pages 81–102 (cited on page 37).
- [Kol65] Andrei N Kolmogorov. “Three approaches to the quantitative definition of information”. In: *Problems of information transmission* 1.1 (1965), pages 1–7 (cited on page 98).
- [Li+04] Ming Li et al. “The similarity metric”. In: *IEEE transactions on Information Theory* 50.12 (2004), pages 3250–3264 (cited on page 257).
- [Llo82] Stuart Lloyd. “Least squares quantization in PCM”. In: *IEEE transactions on information theory* 28.2 (1982), pages 129–137 (cited on page 90).
- [Lot20] Alfred J Lotka. “Analytical note on certain rhythmic relations in organic systems”. In: *Proceedings of the National Academy of Sciences* 6.7 (1920), pages 410–415 (cited on page 37).
- [McM56] Brockway McMillan. “Two inequalities implied by unique decipherability”. In: *IRE Transactions on Information Theory* 2.4 (1956), pages 115–116 (cited on page 89).
- [Pos46] Emil L Post. “A variant of a recursively unsolvable problem”. In: *Bulletin of the American Mathematical Society* 52.4 (1946), pages 264–268 (cited on page 76).
- [QR89] J Ross Quinlan and Ronald L Rivest. “Inferring decision trees using the minimum description length principle”. In: *Information and computation* 80.3 (1989), pages 227–248 (cited on page 233).
- [Shm+10] Galit Shmueli et al. “To explain or to predict?” In: *Statistical science* 25.3 (2010), pages 289–310 (cited on page 233).
- [Sol64] Ray J Solomonoff. “A formal theory of inductive inference. Part I and II”. In: *Information and control* 7.1 (1964), pages 1–22 (cited on page 98).
- [Tur36] Alan Mathison Turing. “On computable numbers, with an application to the Entscheidungsproblem”. In: *J. of Math* 58.345–363 (1936), page 5 (cited on page 75).
- [Tur39] Alan Mathison Turing. “Systems of logic based on ordinals”. In: *Proceedings of the London Mathematical Society* 2.1 (1939), pages 161–228 (cited on page 140).
- [WB68] Chris S Wallace and David M Boulton. “An information measure for classification”. In: *The Computer Journal* 11.2 (1968), pages 185–194 (cited on page 115).
- [WP93] Chris S Wallace and JD Patrick. “Coding decision trees”. In: *Machine Learning* 11.1 (1993), pages 7–22 (cited on page 233).
- [YW09] Ying Yang and Geoffrey I Webb. “Discretization for naive-Bayes learning: managing discretization bias and variance”. In: *Machine learning* 74.1 (2009), pages 39–74 (cited on page 233).





# Index

## Symbols

$\sigma$ -algebra ..... 51

## A

Absolute value ..... 43  
Adjacency matrix ..... 45  
Adjacent vertices ..... 45  
Alphabet ..... 44  
Ancestors of a vertex ..... 46  
Antisymmetric relation ..... 43  
Applicability ..... 169  
Applicability graph ..... 168  
Areas in decay ..... 172  
Average interestingness of an area ..... 172  
Axiomatic interpretation of probability ..... 50

## B

Balanced tree ..... 47  
Bayesian interpretation of probability ..... 50  
Berry paradox ..... 22, 131  
Bijective function ..... 43  
Binary relation ..... 42  
Binary tree ..... 47

Bipartite graph ..... 46  
Blank Symbol ..... 69, 74  
Boston dataset ..... 29  
Branches of a tree ..... 46

## C

Cantor's theorem ..... 20  
Cardinality of a set ..... 42  
Cartesian product ..... 42  
Ceil ..... 43  
Certain event ..... 50  
Characteristic function ..... 43  
Child of a vertex ..... 46  
Classical interpretation of probability ..... 49  
Codomain ..... 43  
Complement of a set ..... 42  
Composition ..... 43  
Computable Function ..... 73  
Computation ..... 71  
Conditional model ..... 134  
Configuration ..... 70  
Configuration yields configuration ..... 70  
Countable many set ..... 43  
Countable set ..... 43  
Cycle ..... 46

**D**

- Degree of a vertex ..... 45  
 Degree sum formula ..... 46  
 Depth of a vertex ..... 46  
 Descendant of a vertex ..... 46  
 Description ..... 23  
 Description function ..... 132  
 Directed graph ..... 45  
 Disjoint sets ..... 42  
 Domain ..... 43  
 Dutch book ..... 50

**E**

- Edges of a graph ..... 45  
 Empty set ..... 42  
 Empty string ..... 44  
 Endpoints of an edge ..... 45  
 Entities ..... 126  
 Entity ..... 20  
 Equal sets ..... 42

**F**

- Field of sets ..... 42  
 Final State ..... 69, 74  
 Floor ..... 43  
 Frequentist interpretation of probability ..... 49  
 full k-ary tree ..... 47  
 Function ..... 43

**G**

- Graph ..... 45, 132

**H**

- Halting problem ..... 72  
 Handshaking theorem ..... 46  
 Height of a tree ..... 46

**I**

- Identity function ..... 43

- Impossible event ..... 50  
 In-degree of a vertex ..... 46  
 Inaccuracy ..... 20, 26, 201  
 Incident edge ..... 45  
 Infinite graph ..... 45  
 Initial State ..... 69, 74  
 Initial vertex ..... 45  
 Injective function ..... 43  
 Input Symbol ..... 69, 74  
 Interdisciplinary new topic ..... 171  
 Interdisciplinary question ..... 170  
 Interestingness of a new topic ..... 171  
 Interestingness of a question ..... 170  
 Interestingness of a topic as a problem ..... 168  
 Interestingness of a topic as a tool ..... 169  
 Intersection of sets ..... 42  
 Intradisciplinary new topic ..... 171  
 Intradisciplinary question ..... 170  
 Inverse function ..... 43  
 Isolated vertex ..... 46

**K**

- k-ary tree ..... 47  
 Known subset of an area ..... 136  
 Kolmogorov's axioms ..... 51

**L**

- Labeled graph ..... 46  
 Leaf vertex ..... 46  
 Leibniz's series ..... 25  
 Length of a string ..... 44  
 Lotka–Volterra differential equation ..... 30

**M**

- Machine State ..... 69, 74  
 Maturity ..... 169  
 Maximum ..... 43  
 Member of a set ..... 42  
 Minimum ..... 43  
 Miscoding ..... 19, 24  
 MNIST ..... 201  
 Model ..... 131  
 Model of a topic ..... 131  
 Multitape Turing machine ..... 70

**N**

- n-tuple ..... 42  
 Neighborhood of a vertex ..... 46  
 Nescience ..... 20, 29  
 Neural network ..... 133  
 New topic ..... 171  
 New topics area ..... 171  
 Newton's second law ..... 26

**O**

- Occam's razor principle ..... 132  
 Oracle Turing machine ..... 21, 25  
 Ordered pair ..... 42  
 Out-degree of a vertex ..... 46  
 Outcome ..... 50  
 Overfitting ..... 202

**P**

- Parent of a vertex ..... 46  
 Pareto optimality ..... 29  
 Partial function ..... 43  
 Partition of a set ..... 42  
 Path ..... 46  
 Pendant vertex ..... 46  
 Perfect conditional model ..... 135  
 Perfect knowledge ..... 31  
 Perfect model ..... 132  
 Perfect model of an area ..... 136  
 Phlogiston ..... 25  
 Power set ..... 42  
 Prefix ..... 44  
 Prefix free set ..... 44  
 Principle of indifference ..... 49  
 Probability ..... 51  
 Probability mass function ..... 52, 55

**Q**

- Question ..... 170

**R**

- Randomness ..... 28  
 Range ..... 43  
 Reflexive relation ..... 42  
 Relevance ..... 167  
 Relevance Graph ..... 167  
 Representation ..... 21  
 Requirements for Interesting Questions ..... 165  
 Research area ..... 136  
 Reverse string ..... 44  
 Root of a tree ..... 46  
 Russel's paradox ..... 20

**S**

- Sample space ..... 50  
 Scientific method ..... 23  
 Self delimited string ..... 44  
 Set ..... 42  
 Set difference ..... 42  
 Set formation notation ..... 42  
 Set of conditional models ..... 134  
 Set of integers ..... 42  
 Set of known research topics ..... 169  
 Set of models of an area ..... 136  
 Set of natural numbers ..... 42  
 Set of positive integers ..... 42  
 Set of positive reals ..... 42  
 Set of rationals ..... 42  
 Set of real numbers ..... 42  
 Set of valid models ..... 132  
 Shortlex ordering ..... 44  
 Sibling to a vertex ..... 46  
 State diagram ..... 69  
 Stock market ..... 203  
 string ..... 44  
 String concatenation ..... 44  
 Subgraph ..... 46  
 Subjective interpretation of probability ..... 50  
 Subset ..... 42  
 Substring ..... 44  
 Subtree ..... 46  
 Surfeit ..... 20, 27  
 Surjective function ..... 43  
 Symmetric relation ..... 43

**T**

- Tape Symbol ..... 69, 74  
Terminal vertex ..... 45  
Total relation ..... 43  
Totally ordered set ..... 43  
Transition Function ..... 69, 74  
Transitive relation ..... 43  
Tree ..... 46  
Trivial model ..... 132  
Turing Machine ..... 68  
Turing machine ..... 21, 23

**U**

- Unbalance dataset ..... 203  
Uncountable set ..... 43  
Union of sets ..... 42  
Universal Turing machine ..... 72  
Universe of a set ..... 42  
Unknown frontier ..... 170  
Unknown unknown ..... 22  
utility function ..... 29

**V**

- Valid models ..... 132  
Venn diagram ..... 42  
Vertices of a graph ..... 45

**W**

- Wallis' product ..... 24  
Weight of an edge ..... 46