BAYESIAN PROGRAMMING

Pierre Bessière
Juan-Manuel Ahuactzin
Kamel Mekhnacha
Emmanuel Mazer

Contact:

Pierre.Bessiere@imag.fr

Juan-Manuel.Ahuactzin@inrialpes.fr

To the late Edward T. Jaynes

for his doubts about certitudes

and for his certitudes about probabilities

Table of content

1

Introduction

The most incomprehensible thing about the world is that it is comprehensible

Albert Einstein

1.1 Probability an alternative to logic

Computers have brought a new dimension to modeling. A model, once translated into a program and run on a computer, may be used to understand, measure, simulate, mimic, optimize, predict, and control. During the last fifty years science, industry, finance, medicine, entertainment, transport, and communication have been completely transformed by this revolution.

However, models and programs suffer from a fundamental flaw: *incompleteness*. Any model of a real phenomenon is incomplete. Hidden variables, not taken into account in the model, influence the phenomenon. The effect of the hidden variables is that the model and the phenomenon never have the exact same behaviors. *Uncertainty* is the direct and unavoidable consequence of incompleteness. A model may not foresee exactly the future

observations of a phenomenon, as these observations are biased by the hidden variables, and may not predict the consequences of its decisions exactly.

Computing a cost price to decide on a sell price may seem a purely arithmetic operation consisting of adding elementary costs. However, often these elementary costs may not be known exactly. For instance, a part's cost may be biased by exchange rates, production cost may be biased by the number of orders and transportation costs may be biased by the period of the year. Exchange rates, the number of orders, and the period of the year when unknown, are hidden variables, which induce uncertainty in the computation of the cost price.

Analyzing the content of an email to filter spam is a difficult task, because no word or combination of words can give you an absolute certitude about the nature of the email. At most, the presence of certain words is a strong clue that an email is a spam. It may never be a conclusive proof, because the context may completely change its meaning. For instance, if one of your friends is forwarding you a spam for discussion about the spam phenomenon, its whole content is suddenly not spam any longer. A linguistic model of spam is irremediably incomplete because of this boundless contextual information. Filtering spam is not hopeless and some very efficient solution exists, but the perfect result is a chimera.

Machine control and dysfunction diagnosis is very important to industry. However, the dream of building a complete model of a machine and all its possible failures is an illusion. One should recall the first "bug" of the computer era: the moth located in relay 70 panel F of the Harvard Mark II computer. Once again, it does not mean that control and diagnosis is hopeless, it only means that models of these machines should take into account their own incompleteness and the resulting uncertainty.

In 1781, Sir William Herschell discovered Uranus, the seventh planet of the solar system. In 1846, Johann Galle observed for the first time, Neptune, the eighth planet. In the meantime, both Urbain Leverrier, a French astronomer, and John Adams, an English one, became interested in the "uncertain" trajectory of Uranus. The planet was not following exactly the trajectory that Newton's theory of gravitation predicted. They both came to the conclusion that these irregularities should be the result of a hidden variable not taken into account by the model: the existence of an eighth planet. They even went much further, finding the most probable position of this eighth planet. The Berlin observatory received Leverrier's prediction on September 23, 1846 and Galle observed Neptune the

very same day!

Logic is both the mathematical foundation of rational reasoning and the fundamental principle of present day computing. However, logic, by essence, is restricted to problems where information is both *complete* and *certain*. An *alternative mathematical framework* and an *alternative computing framework* are both needed to deal with incompleteness and uncertainty.

Probability theory is this alternative mathematical framework. It is a model of rational reasoning in the presence of incompleteness and uncertainty. It is an extension of logic where both certain and uncertain information have their places.

James C. Maxwell stated this point synthetically:

> *The actual science of logic is conversant at present only with things either certain, impossible, or entirely doubtful, none of which (fortunately) we have to reason on. Therefore the true logic for this world is the calculus of Probabilities, which takes account of the magnitude of the probability which is, or ought to be, in a reasonable man's mind.*
>
> *James C. Maxwell; quote in "Probability Theory - The Logic of Science" by Edward T. Jaynes (*Jaynes, 2003*)*

Considering probability as *a model of reasoning* is called the *subjectivist* or *Bayesian* approach. It is opposed to the *objectivist* approach, which considers probability as *a model of the world*. This opposition is not only an epistemological controversy; it has many fundamental and practical consequences[1].

To model reasoning, you must take into account the *preliminary knowledge* of the subject who is doing the reasoning. This preliminary knowledge plays the same role as the axioms in logic. Starting from different preliminary knowledge may lead to different conclusions. Starting from wrong preliminary knowledge will lead to wrong conclusions even with perfectly correct reasoning. Reaching wrong conclusions following correct reasoning proves that the preliminary knowledge was wrong, offers the opportunity to correct it and eventually leads you to learning. Incompleteness is simply the irreducible gap between the preliminary knowledge and the phenomenon and uncertainty is a direct

1. See “Subjectivism vs objectivism controversy”, page 203

and measurable consequence of this imperfection.

In contrast, modeling the world by denying the existence of a "subject" and consequently rejecting preliminary knowledge leads to complicated situations and apparent paradoxes. This rejection implies that if the conclusions are wrong, either the reasoning could be wrong or the data could be aberrant, leaving no room for improvement or learning. Incompleteness does not mean anything without preliminary knowledge, and uncertainty and noise must be mysterious properties of the physical world.

The objectivist school has been dominant during the 20th century, but the subjectivist approach has a history as long as probability itself. It can be traced back to Jakob Bernoulli in 1713:

> *Uncertainty is not in things but in our head: uncertainty is a lack of knowledge.*
>
> *Jakob Bernoulli, Ars Conjectandi (*Bernouilli, 1713*);*

to the Marquis Simon de Laplace[1], one century later, in 1812:

> *Probability theory is nothing but common sense reduced to calculation.*
>
> *Simon de Laplace, Théorie Analytique des Probabilités (*Laplace, 1812*)*

to the already quoted James C. Maxwell in 1850 and to the visionary Henri Poincaré in 1902:

> *Randomness is just the measure of our ignorance.*
>
> *To undertake any probability calculation, and even for this calculation to have a meaning, we have to admit, as a starting point, an hypothesis or a convention, that always comprises a certain amount of arbitrariness. In the choice of this convention, we can be guided only by the principle of sufficient reason.*
>
> *From this point of view, every sciences would just be unconscious applications of the calculus of probabilities. Condemning this calculus would*

1. See “LAPLACE, MARQUIS SIMON DE (WHO IS?)”, page 203

be condemning the whole science.

*Henri Poincaré, La science et l'hypothèse (*Poincaré, 1902*)*

and finally, by Edward T. Jaynes[1] in his book Probability theory: the logic of science (Jaynes, 2003) where he brilliantly presents the subjectivist alternative and sets clearly and simply the basis of the approach:

By inference we mean simply: deductive reasoning whenever enough information is at hand to permit it; inductive or probabilistic reasoning when - as is almost invariably the case in real problems - all the necessary information is not available. Thus the topic of "Probability as Logic" is the optimal processing of uncertain and incomplete knowledge.

*Edward T. Jaynes, Probability Theory: The Logic of Science (*Jaynes, 2003*)*

1.2 A need for a new computing paradigm

Bayesian probability theory is clearly the sought mathematical alternative to logic[2].

However, we want working solutions to incomplete and uncertain problems. Consequently, we require an alternative computing framework based on Bayesian probabilities.

To create such a complete computing Bayesian framework, we reqire a *new modeling methodology* to build probabilistic models, we require *new inference algorithms* to automate probabilistic calculus, we require *new programming languages* to implement these models on computers, and finally, we will eventually require *new hardware* to run these Bayesian programs efficiently.

The ultimate goal is a *Bayesian computer*. The purpose of this book is to describe the current first steps in this direction.

1.3 A need for a new modeling methodology

The existence of a systematic and generic method to build models is a *sine qua non* requirement for the success of a modeling and computing paradigm. This is why algo-

1. See "JAYNES, ED. T. (WHO IS?)", page 203
2. See "Cox theorem (What is?)", page 202

rithms are taught in the basic course of computer science giving students the basic and necessary methods to develop classical programs. Such a systematic and generic method exists within the Bayesian framework. Moreover, this method is very simple even if it is atypical and a bit worrisome at the beginning.

The purpose of Chapters 2 to 11 is to present this new modeling methodology. The presentation is intended for the general public and does not suppose any prerequisites other than a basic fundation in mathematics. Its purpose is to introduce the fundamental concepts, to present the novelty and interest of the approach, and to initiate the reader to the subtle art of Bayesian modeling. Numerous simple examples of applications are presented in different fields such as $$$ medicine, robotics, finance, and process control.

Chapter 2 gently introduces the basic concepts of Bayesian Programming. We start with a simple example of a Bayesian spam filter that helps you dispose of junk emails. Commercially available software is based on a similar approach.

The problem is very easy to formulate. We want to classify texts (email) in one of two categories either "spam" or "to consider". The only information we have to classify the emails is their content: a set of words.

The classifier should furthermore be able to adapt to its user and to learn from experience. Starting from an initial standard setting, the classifier should modify its internal parameters when the choice of the user does not correspond to its own decision. It will hence adapt to a user's criteria to choose between "spam" and "not-spam". It will improve its results as it analyzes increasingly classified emails.

The goal of Chapter 3 is to explore thoroughly the concept of incompleteness with a very simple physical experiment called the "beam in the bin" experiment. We demonstrate how incompleteness is the source of uncertainty by using a second, more elaborate experiment involving a model of a water treatment center.

Two sources of uncertainty are shown: the existence of hidden variables and the inaccuracy of measures. The effects of both are quantified. We also demonstrate that ignoring incompleteness may lead to certain but definitely wrong decisions. Finally, we explain how learning transforms incompleteness into uncertainty and how probabilistic inference leads to informed decision despite this uncertainty.

In Chapter 4 we present in some detail the fundamental notion of *description*. A descrip-

tion is the probabilistic model of a given phenomenon. It is obtained after two phases of development: first, a *specification* phase where the programmer expresses his own knowledge about the modeled phenomenon in probabilistic terms; and second, an *identification* phase where this starting probabilistic canvas is refined by learning from data. Descriptions are the basic elements that are used, combined, composed, manipulated, computed, compiled, and questioned in different ways to build Bayesian programs.

The specification itself is obtained in three phases. First, programmers must choose the pertinent *variables*. Second, in a *decomposition* phase they must express the joint probability on the selected variables as a product of simpler distributions. Finally, they must choose a parametric *form* for each of these distributions.

These three phases are depicted by: (i) a robotic example where a small mobile robot is taught how to push, avoid, and circle small objects and (ii) a continuation of the water treatment center instance where the corresponding description is illustrated.

Two "equations" may summarize the content of this chapter: "Description = Specification + Identification" and "Specification = Variables + Decomposition + Forms".

In Chapter 5 the notions of independence and conditional independence are introduced. We demonstrate the importance of conditional independence in actually solving and computing complex probabilistic problems.

The water treatment center instance is further developed to exemplify these central concepts.

The water treatment center instance is completed in Chapter 6. The probabilistic description of this process (built in Chapters 4 and 5) is used to solve different problems: prediction of the output, choice of the best control strategy, and diagnosis of failures. This shows that multiple questions may be asked of the same description to solve very different problems. This clear separation between the model and its use is a very important feature of Bayesian Programming.

This chapter completes the introduction and definition of a *Bayesian program,* which is made of both a *description* and a *question*: "Bayesian Program = Description + Question".

The essence of the critical computational difficulties of Bayesian inference is presented but, *per contra*, we explain that in Bayesian modeling there are neither "ill posed problems" nor opposition between "direct" and "inverse" problems. Within the Bayesian

framework, any inverse problem has an evident theoretical solution, but this solution may be very costly to compute.

Chapters 2 to 6 present the concept of the Bayesian program. Chapters 8 to 13 are used to show how to combine elementary Bayesian programs to build more complex ones. Some analogies are stressed between this probabilistic mechanism and the corresponding algorithmic ones, for instance the use of subroutines or conditional and case operators.

For instance, Chapter 8 shows how Bayesian subroutines can be called within Bayesian programs. As for in classical programming, subroutines are a major means to build complex models from simpler ones, using them as elementary bricks. We can create a hierarchy of probabilistic descriptions resulting from either a top-down analysis or a bottom-up elaboration.

A financial analysis example is used to exemplify this concept. $$$more to come when Chapter 7 will be written$$$.

Chapter 9 introduces the Bayesian program combination. Using both a simple robotic example and a more elaborate natural risk analysis problem we show how to combine different Bayesian programs with the help of a choice variable. If known with certainty, this choice variable would act as a switch between the different models. In that case this model acts as an "if-then-else" or a case statement. If the choice variable is not known with certainty, we then obtain the probabilistic equivalent of these conditional constructors.

In Chapter 7 we investigate how to sequence Bayesian programs. *Inverse programming* is proposed as a potentially efficient solution. It consists of expressing independently the conditional probabilities of the conditions knowing the action. Even in atypical cases, this modeling method appears to be convenient and generic. Furthermore it leads to very easy learning schemes.

The inverse programming concept is exemplified with a video game application. We address the problem of real-time reactive selection of elementary behaviors for an agent playing a first person shooter game. We show how Bayesian Programming leads to a more condensed and easier formalization than a finite state machine. We also demonstrate that using this technique it is easy to implement learning by imitation in a fully transparent way for the player.

Chapter 10 approaches the topic of Bayesian program iteration. *Recursive Bayesian estimation* and more specifically *Bayesian filtering* are presented as the main tools to implement iteration.

An Advanced Driver Assistance System (ADAS) application is presented as an example. The goal of ADAS systems is largely to automate driving by assisting and eventually replacing the human driver in some tasks.

Chapter 11 explains how to combine probabilistic inference and algebraic operations. A very simple example of computing a cost price for container chartering is presented. The global cost price is the sum of sub costs, some of which can only be known approximately. We show how the probability distribution on the global cost may be derived from the different probability distributions on the subcosts.

Bayesian Programming may be intimately combined with classical programming. It is possible to incorporate Bayesian computation within classical programs. Moreover, it is also possible to use classical programming within Bayesian programs. Indeed, there are different ways to call functions inside Bayesian programs, and these are presented in Chapter 11.

A Computer Aided Design (CAD) application is used to demostrate this functionality.

1.4 A need for new inference algorithms

A modeling methodology is not sufficient to run Bayesian programs. We also require an efficient Bayesian inference engine to automate the probabilistic calculus. This assumes we have a collection of inference algorithms adapted and tuned to more or less specific models and a software architecture to combine them in a coherent and unique tool.

Numerous such Bayesian inference algorithms have been proposed in the litterature. The purpose of this book is not to present this different computing technique and its associated models once more. Instead, we offer a synthesis of this work and a number of bibliographic references for those who would like more detail on these subjects.

The purpose of Chapter 12 is to present Bayesian Programming formally. It may seem weird to present the formalism near the end of the book and after all the examples. We have made this choice to help comprehension and favor intuition without sacrificing rigor. Anyone can check after reading this chapter that all the examples and programs

presented beforehand comply with the formalism.

The goal of Chapter 13 is to revisit the main existing Bayesian models. We use the Bayesian Programming formalism to present them systematically. This is a good way to be precise and concise and it also simplifies their comparison.

We chose to divide the different probabilistic models into two categories, the general purpose probabilistic models and the problem oriented probabilistic models.

In the general purpose category, the modeling choices are made independently of any specific knowledge about the modeled phenomenon. Most commonly these choices are essentially made to keep inferences tractable. However, the technical simplifications of these models may be compatible with large classes of problems and consequently may have numerous applications. Among others, we are restating in the Bayesian Programming (BP) formalism Bayesian Networks (BN), Dynamic Bayesian Networks (DBN), Hidden Markov Models (HMM), Kalman filters, and mixture models.

In the problem oriented category, on the contrary, the modeling choices and simplifications are decided according to some specific knowledge about the modeled phenomenon. These choices could eventually lead to very poor models from a computational point of view. However, most of the time problem dependent knowledge, such as conditional independence between variables, leads to very important and effective simplifications and computational improvements.

Chapter 16 surveys the main available general purpose algorithms for Bayesian inference.

It is well known that general Bayesian inference is a very difficult problem, which may be practically intractable. Exact inference has been proved to be NP-hard (Cooper, 1990) as has the general problem of approximate inference (Dagum & Luby, 1993).

Numerous heuristics and restrictions to the generality of possible inferences have been proposed to achieve admissible computation time. The purpose of this chapter is to make a short review of these heuristics and techniques.

Before starting to crunch numbers, it is usually possible (and wise) to make some symbolic computations to reduce the amount of numerical computation required. The first section of this chapter presents the different possibilities. We will see that these symbolic computations can be either exact or approximate.

Once simplified, the expression obtained must be numerically evaluated. In a few

cases exact (exhaustive) computation may be possible thanks to the previous symbolic simplification, but most of the time, even with the simplifications, only approximate calculations are possible. The second section of this chapter describes the main algorithms to do so.

Finally, Chapter 15 surveys the different learning algorithms. The best known ones are rapidly recalled and restated in Bayesian Programming terms. $$$More to come when Chapter 18 has been written$$$.

1.5 A need for a new programming language and new hardware

A modeling methodology and new inference algorithms are not sufficient to make these models operational. We also require new programming languages to implement them on classical computers and, eventually, new specialized hardware architectures to run these programs efficiently.

However captivating these topics may be, we chose not to deal with them in this book.

Concerning new programming languages, this book comes with a companion website (Bayesian-Programming.org) where such a programming language called ProBT® is provided under a free license restricted to noncommercial uses.

ProBT® is a C++ multi-platform professional library used to automate probabilistic calculus. The ProBT® library has two main components: (i) a friendly Application Program Interface (API) for building Bayesian models and (ii) a high-performance Bayesian Inference Engine (BIE) allowing the entire probability calculus to be executed exactly or approximately.

ProBT® comes with its complete documentation and numerous examples, including those used in this book. Utilization of the library requires some computer science proficiency, which is not required from the readers of this book. This companion website is a plus but is not necessary in the comprehension of this book.

It is too early to be clear about new hardware dedicated to probabilistic inference, and the book is already too long to make room for one more topic!

However, we would like to stress that 25 years ago, no one dared to dream about graphical computers. Today, no one dares to sell a computer without a graphical display

with millions of pixels able to present realtime 3D animations or to play high quality movies thanks to specific hardware that makes such a marvel feasible.

We are convinced that 25 years from now, the ability to treat incomplete and uncertain data will be as inescapable for computers as graphical abilities are today. We hope that you will also be convinced of this at the end of this book. Consequently, we will require specific hardware to face the huge computing burden that some Bayesian inference problems may generate.

Many possible directions of research may be envisioned to develop such new hardware. Some, especially promising, are inspired by biology. Indeed, some researchers are currently exploring the hypothesis that the central nervous system (CNS) could be a probabilistic machine either at the level of individual neurons or assemblies of neurons. Feedback from these studies could provide inspiration for this necessary hardware.

1.6 A place for numerous controversies

We believe that Bayesian modeling is an elegant matter that can be presented simply, intuitively, and with mathematical rigor. We hope that we succeed in doing so in this book. However, the subjectivist approach to probability has been and still is a subject of countless controversies.

Some questions must be asked, discussed, and answered, such as: the role of decision theory; the dilemma of bias versus variance; the computational complexity of Bayesian inference; the frequentist versus nonfrequentist argument; fuzzy logic versus the probabilistic treatment of uncertainty; physical causality versus logical causality; and last but not least, the subjectivist versus objectivist epistemological conception of probability itself.

To make the main exposition as clear and simple as possible, none of these controversies, historical notes, epistemological debates, and tricky technical questions are discussed in the body of the book. We have made the didactic choice to develop all these questions in a special annex entitled "FAQ and FAM" (Frequently Asked Questions and Frequently Argued Matters).

This annex is organized as a collection of "record cards", four pages long at most, presented in alphabetical order. Cross references to these subjects are included in the main text for readers interested in going further than a simple presentation of the principles of Bayesian modeling.

2

Basic Concepts

> *Far better an approximate answer to the right question which is often vague, than an exact answer to the wrong question which can always be made precise.*
>
> *John W. Tuckey*

The purpose of this chapter is to gently introduce the basic concepts of Bayesian Programming. These concepts will be extensively used and developed in Chapters 4 to 13 and they will be revisited, summarized and formally defined in Chapter 12.

We start with a simple example of Bayesian spam filtering, which helps to eliminate junk emails. Commercially available software is based on a similar approach.

The problem is very easy to formulate. We want to classify texts (email) into one of two categories either “nonspam” or “spam”. The only information we can use to classify the emails is their content: a set of words.

The classifier should furthermore be able to adapt to its user and to learn from experience. Starting from an initial standard setting, the classifier should modify its internal parameters when user disagrees with its own decision. It will hence adapt to users’ crite-

ria to choose between “nonspam” and “spam”. It will improve its results as it encounters increasingly classified emails. The classifier uses an N word dictionary. Each email will be classified according to the presence or absence of each of the words in the dictionary.

2.1 Variable

The variables necessary to write this program are the following:

1. $Spam$ [1]: a binary variable, false if the email is not spam and true otherwise.

2. $W_0, W_1, ..., W_{N-1}$: N binary variables. W_n is true if the n^{th} word of the dictionary is present in the text.

These N + 1 binary variables sum up all the information we have about an email.

2.2 Probability

A variable can have one and only one value at a given time, so the value of $Spam$ is either true [2] or false , as the email may either be spam or not.

However, this value may be unknown. Unknown does not mean that you do not have any information concerning $Spam$. For instance, you may know that the average rate of nonspam email is 25%. This information may be formalized, writing:

1. $\mathbf{P}([Spam = \text{false}]) = 0{,}25$, which stands for “the probability that an email is not spam is 25%”

2. $\mathbf{P}([Spam = \text{true}]) = 0{,}75$

2.3 The normalization postulate

According to our hypothesis, an email is either interesting to read or spam. It means that it cannot be both but it is necessarily one of them. This implies that:

1. Variables will be denoted by their name in italics with initial capital.
2. Variable values will be denoted by their name in roman, in lowercase.

$$\mathbf{P}([Spam = \text{false}]) + \mathbf{P}([Spam = \text{true}]) = 1{,}0 \tag{2.1}$$

This property is true for any discrete variable (not only for binary ones) and consequently the probability distribution on a given variable X should necessarily be normalized:

$$\sum_{x \in X} \mathbf{P}([X = x]) = 1{,}0 \tag{2.2}$$

For the sake of simplicity, we will use the following notation

$$\sum_{X} \mathbf{P}(X) = 1{,}0 \tag{2.3}$$

2.4 Conditional probability

We may be interested in the probability that a given variable assumes a value based on some information. This is called a conditional probability.

For instance, we may be interested in the probability that a given word appears in spam: $\mathbf{P}([W_j = \text{true}] \,|\, [Spam = \text{true}])$. The sign " | " separates the variables into two sets: on the right are the variables with values known with certainty, on the left the probed variables.

This notation may be generalized as: $\mathbf{P}(W_j \,|\, [Spam = \text{true}])$, which stands for the probability distribution on W_j knowing that the email is spam. This distribution is defined by two probabilities corresponding to the two possible values of W_j. For instance:

1. $\mathbf{P}([W_j = \text{false}] \,|\, [Spam = \text{true}]) = 0{,}9996$

2. $\mathbf{P}([W_j = \text{true}] \,|\, [Spam = \text{true}]) = 0{,}0004$

Analogously to Expression (2.3) we have that for any two variables X and Y

$$\sum_{X} \mathbf{P}(X \mid Y) = 1{,}0 \tag{2.4}$$

Consequently, $\sum_{W_j} \mathbf{P}(W_j \mid [Spam = \text{true}]) = 1{,}0$.

2.5 Variable conjunction

We may also be interested in the probability of the conjunction of two variables: $\mathbf{P}(Spam \wedge W_j)$.

$Spam \wedge W_j$, the conjunction of the two variables $Spam$ and W_j, is a new variable that can take four different values:

$$\{(\text{false}, \text{false}), (\text{false}, \text{true}), (\text{true}, \text{false}), (\text{true}, \text{true})\} \tag{2.5}$$

This may be generalized as the conjunction of an arbitrary number of variables. For instance, in the sequel, we will be very interested in the joint probability distribution of the conjunction of N + 1 variables:

$$\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}) \tag{2.6}$$

2.6 The conjunction postulate (Bayes theorem)

The probability of a conjunction of two variables X and Y may be computed according to the Conjunction Rule:

$$\begin{aligned} \mathbf{P}(X \wedge Y) &= \mathbf{P}(X) \times \mathbf{P}(Y \mid X) \\ &= \mathbf{P}(Y) \times \mathbf{P}(X \mid Y) \end{aligned} \tag{2.7}$$

This rule is betterl known under the form of the so called Bayes theorem:

$$\mathbf{P}(X \mid Y) = \frac{\mathbf{P}(X) \times \mathbf{P}(Y \mid X)}{\mathbf{P}(Y)} \tag{2.8}$$

However, we prefer the first form, which clearly states that it is a means of computing the probability of a conjunction of variables according to both the probabilities of these variables and their relative conditional probabilities.

For instance, we have:

$$\begin{aligned}\mathbf{P}([Spam = \text{true}] \wedge [W_j = \text{true}]) &= \mathbf{P}([Spam = \text{true}]) \times \mathbf{P}([W_j = \text{true}] \,|\, [Spam = \text{true}]) \\ &= 0{,}75 \times 0{,}0004 \\ &= 0{,}0003 \\ &= \mathbf{P}([W_j = \text{true}]) \times \mathbf{P}([Spam = \text{true}] \,|\, [W_j = \text{true}])\end{aligned} \tag{2.9}$$

2.7 Syllogisms

It is very important to acquire a clear intuitive feeling of what a conditional probability and the Conjunction Rule mean. A first step toward this understanding may be to restate the classical logical syllogisms in their probabilistic forms.

Let us first recall the two logical syllogisms:

1. Modus Ponens: $a \wedge [a \Rightarrow b] \rightarrow b$ [1]
   if a is true and if a implies b then b is true

2. Modus Tollens: $\neg b \wedge [a \Rightarrow b] \rightarrow \neg a$
   if b is false and if a implies b then a is false

For instance, if a stands for "n may be divided by 9" and b stands for "n may be divided by 3", we know that $a \Rightarrow b$, and we have:

1. Modus Ponens: If "n may be divided by 9" then "n may be divided by 3".

2. Modus Tollens: if "n may be divided by 3" is false then "n may be divided by 9" is also false.

Using probabilities, we may state:

1. Modus Ponens: $\mathbf{P}(b \,|\, a) = 1$, which means that knowing that a is true then we may be sure that b is true.

2. Modus Tollens: $\mathbf{P}(\neg a \,|\, \neg b) = 1$, which means that knowing that b is false then we may be sure that a is false.

1. Logical propositions will be denoted by name in italics, in lowercase.

$\mathbf{P}(\neg a \mid \neg b) = 1$ may be derived from $\mathbf{P}(b \mid a) = 1$, using the normalization and conjunction postulates:

$$
\begin{aligned}
\mathbf{P}(\neg a \mid \neg b) &= 1{,}0 - \mathbf{P}(a \mid \neg b) && \text{from (2.3)} \\
&= 1{,}0 - \frac{\mathbf{P}(\neg b \mid a) \times \mathbf{P}(a)}{\mathbf{P}(\neg b)} && \text{from (2.8)} \\
&= 1{,}0 - \frac{(1 - \mathbf{P}(b \mid a)) \times \mathbf{P}(a)}{\mathbf{P}(\neg b)} && \text{from (2.3)} \\
&= 1{,}0 && \text{because } \mathbf{P}(b \mid a) = 1{,}0
\end{aligned}
$$

However, using probabilities we may go further than with logic:

1. From $\mathbf{P}(b \mid a) = 1$, using normalization and conjunction postulates we may derive[1] that $\mathbf{P}(a \mid b) \geq \mathbf{P}(a)$, which means that if we know that b is true, the probability that a is true is higher than it would be if we knew nothing about b.
   Obviously, the probability that "n may be divided by 9" is higher if you do know that "n may be divided by 3" than if you do not. This very common reasoning which is beyond the scope of pure logic but is very simple in the Bayesian framework.

2. From $\mathbf{P}(b \mid a) = 1$, using the normalization and conjunction postulates we may derive[2] that $\mathbf{P}(\neg b \mid \neg a) \leq \mathbf{P}(\neg b)$, which means that if we know that a is false the probability that b is false is less than it would be if we knew nothing about a.
   The probability that "n may be divided by 3" is less if you know that n may not be divided by 9 than if you do not know anything about n.

2.8 The marginalization rule

A very useful rule, called the marginalization rule, may be derived from the normalization and conjunction postulates. This rule states:

1. This derivation is a good exercise. The solution may be found in the chapter 12 at §
2. This derivation is a good exercise. The solution may be found in the chapter $$$ at § $$$

$$\sum_X \mathbf{P}(X \wedge Y) = \mathbf{P}(Y) \tag{2.10}$$

It may be derived as follows:

$$\begin{aligned} \sum_X \mathbf{P}(X \wedge Y) &= \sum_X \mathbf{P}(Y) \times \mathbf{P}(X \mid Y) && \text{from (2.7)} \\ &= \mathbf{P}(Y) \times \sum_X \mathbf{P}(X \mid Y) \\ &= \mathbf{P}(Y) && \text{from (2.4)} \end{aligned} \tag{2.11}$$

2.9 Joint distribution and questions

The joint distribution on a set of two variables X and Y is the distribution on their conjunction: $\mathbf{P}(X \wedge Y)$. If you know the joint distribution, then you know everything you may want to know about the variables. Indeed, using the conjunction and marginalization rules you have:

1. $\mathbf{P}(Y) = \sum_X \mathbf{P}(X \wedge Y)$

2. $\mathbf{P}(X) = \sum_Y \mathbf{P}(X \wedge Y)$

3. $\mathbf{P}(Y \mid X) = \dfrac{\mathbf{P}(X \wedge Y)}{\sum_Y \mathbf{P}(X \wedge Y)}$

4. $\mathbf{P}(X \mid Y) = \dfrac{\mathbf{P}(X \wedge Y)}{\sum_X \mathbf{P}(X \wedge Y)}$

This is of course also true for a joint distribution on more than two variables.

For our spam instance, if you know the joint distribution:

$$\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}) \tag{2.12}$$

you can compute any of $3^{(N+1)} - 2^{(N+1)}$ possible *questions* that you can imagine on this set of N + 1 variables.

A question is defined by partitionning a set of variables in three subsets: the searched variables (on the left of the conditioning bar), the known variables (on the right of the conditioning bar) and the free variables. The searched variables set must not be empty.

Examples of these questions are:

1. $\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})$, the joint distribution itself;

2. $\mathbf{P}(Spam) = \sum_{W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}} \mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})$,

   the a priori probability to be spam;

3. $\mathbf{P}(W_j) = \sum_{Spam \wedge W_0 \wedge ... \wedge W_{j-1} \wedge W_{j+1} \wedge ... \wedge W_{N-1}} \mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})$,

   the a priori probability for the j^{th} word of the dictionary to appear;

4. $\mathbf{P}(W_j \mid [Spam = \text{true}]) = \dfrac{\sum_{W_1 \wedge ... \wedge W_{j-1} \wedge W_{j+1} \wedge ... \wedge W_N} \mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}{\sum_{W_1 \wedge ... \wedge W_N} \mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}$

   the probability for the j^{th}, word to appear, knowing that the text is a spam;

5. $\mathbf{P}(Spam \mid [W_j = \text{true}]) = \dfrac{\sum_{W_0 \wedge ... \wedge W_{j-1} \wedge W_{j+1} \wedge ... \wedge W_{N-1}} \mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}{\sum_{Spam \wedge W_0 \wedge ... \wedge W_{j-1} \wedge W_{j+1} \wedge ... \wedge W_{N-1}} \mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}$

   the probability for the email to be spam knowing that the j^{th} word appears in the text;

6. $\mathbf{P}(Spam \mid W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}) = \dfrac{\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}{\sum_{Spam} \mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}$

finally, the most interesting one, the probability that the email is spam knowing for all N words in the dictionary if they are present or not in the text.

2.10 Decomposition

The key challenge for a Bayesian programmer is to specify a way to compute the joint distribution that has the three main qualities of being a *good model*, *easy to compute* and *easy to identify*.

This is done using a *decomposition* that restates the joint distribution as a product of simpler distributions.

Starting from the joint distribution and applying recursively the conjunction rule we obtain:

$$\begin{aligned}&\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})\\ &= \mathbf{P}(Spam) \times \mathbf{P}(W_0 \mid Spam) \times \mathbf{P}(W_1 \mid Spam \wedge W_0)\\ &\times ... \times \mathbf{P}(W_{N-1} \mid Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-2})\end{aligned} \tag{2.13}$$

This is an exact mathematical expression.

We simplify it drastically by assuming that the probability of appearance of a word knowing the nature of the text (spam or not) is independent of the appearance of the other words. For instance, we assume that:

$$\mathbf{P}(W_1 \mid Spam \wedge W_0) = \mathbf{P}(W_1 \mid Spam) \tag{2.14}$$

We finally obtain:

$$\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}) = \mathbf{P}(Spam) \times \prod_{i=0}^{N-1} \mathbf{P}(W_i \mid Spam) \tag{2.15}$$

Figure 2.1 shows the graphical model of this expression.

Observe that the assumption of independence between words is clearly not completely true. For instance, it completely neglects that the appearance of pairs of words may be more significant than isolated appearances. However, as subjectivists, we assume this hypothesis and may develop the model and the associated inferences to test how reliable it is.

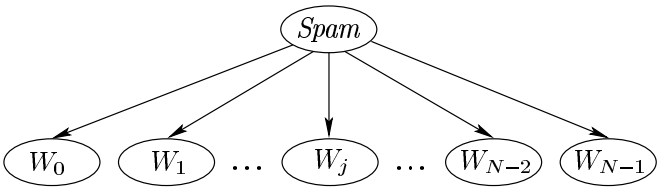

Figure 2.1: The graphical model of the Bayesian spam filter.

2.11 Parametric forms

To be able to compute the joint distribution, we must now specify the N + 1 distributions appearing in the decomposition.

We already specified $\mathbf{P}(Spam)$ in Section 2.2:

- $\mathbf{P}(Spam)$:
  - $\mathbf{P}([Spam = \text{false}]) = 0{,}25$
  - $\mathbf{P}([Spam = \text{true}]) = 0{,}75$

Each of the N forms $\mathbf{P}(W_i \mid Spam)$ must in turn be specified. The first idea is to simply count the number of times the i^{th} word of the dictionary appears in both spam and nonspam. This would naively lead to histograms:

- $\mathbf{P}(W_i \mid Spam)$:
  - $\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{false}]) = \frac{n_f^i}{n_f}$
  - $\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{true}]) = \frac{n_t^i}{n_t}$

where, n_f^i stands for the number of appearances of the i^{th} word in nonspam emails and n_f stands for the total number nonspam emails. Similarly, n_t^i stands for the number of appearences of the i^{th} word in spam emails and n_t stands for the total number of spam emails.

The drawback of histograms is that when no observation has been made, the probabilities are null. For instance, if the i^{th} word has never been observed in spam then:

$$\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{true}]) = 0{,}0 \tag{2.16}$$

A very strong assumption indeed, which says that what has not yet been observed is impossible! Consequently, we prefer to assume that the parametric forms $\mathbf{P}(W_i \mid Spam)$ are Laplace succession laws rather than histograms:

- $\mathbf{P}(W_i \mid Spam)$:

  - $\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{false}]) = \dfrac{1 + n_f^i}{\mathbf{CARD}(W_i) + n_f}$

  - $\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{true}]) = \dfrac{1 + n_t^i}{\mathbf{CARD}(W_i) + n_t}$

where $\mathbf{CARD}(W_i)$ stands for the number of possible values of variable W_i. Here, $\mathbf{CARD}(W_i) = 2$ as W_i is a binary variable.

If the i^{th} word has never been observed in spam then:

$$\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{true}]) = \frac{1}{2 + n_t} \tag{2.17}$$

which tends toward zero when n_t tends toward infinity but never equals zero. An event not yet observed is not completely impossible, even if it becomes very improbable if it has never been observed in a long series of experiments.

2.12 Identification

The N forms $\mathbf{P}(W_i \mid Spam)$ are not yet completely specified because the 2N + 2 parameters $n_f^{i = 0, 2, ..., N-1}$, $n_t^{i = 0, 2, ..., N-1}$, n_f and n_t have no values.

The identification of these parameters could be done either by batch processing of a series of classified emails or by an incremental updating of the parameters using the user's classification of the email as they arrive.

Both methods could be combined: the system could start with initial standard values of these parameters issued from a generic database, then some incremental learning customizes the classifier to each individual user.

2.13 Specification = Variables + Decomposition + Parametric Forms

We call *specification* the part of the Bayesian program specified by the programmer. This part is always made of the same three subparts:

1. *Variables*: the choice of the relevant variables for the problem.

2. *Decomposition*: the expression of the joint probability distribution as the product of simpler distributions.

3. *Parametric Forms*: the choice of the mathematical functions forms of each of these distributions.

2.14 Description = Specification + Identification

We call the *description* the probabilistic model of our problem. The description is the joint probability distribution on the relevant variables. It is completely specified when the eventual free parameters of the *specification* are given values after an *identification* phase.

2.15 Question

Once you have a description (a way to compute the joint distribution), it is possible to ask any *questions,* as we saw in § 2.9.

For instance, after some simplification, the answers to our six questions are:

1. $\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}) = \mathbf{P}(Spam) \times \prod_{i=0}^{N-1} \mathbf{P}(W_i \mid Spam)$

   By definition, the joint distribution is equal to the decomposition.

2. $\mathbf{P}(Spam) = \mathbf{P}(Spam)$

   as $\mathbf{P}(Spam)$ appears as such in the decomposition.

3. $\mathbf{P}(W_j) = \sum_{Spam} \mathbf{P}(Spam) \times \mathbf{P}(W_j \mid Spam)$

the a priori probability for the j^{th} word of the dictionary to appear, which gives

$$\mathbf{P}([W_j = \text{true}]) = \left(0{,}25 \times \frac{1 + n_f^j}{2 + n_f}\right) + \left(0{,}75 \times \frac{1 + n_t^j}{2 + n_t}\right)$$

4. $\mathbf{P}(W_j \mid [Spam = \text{true}]) = \frac{1 + n_t^j}{2 + n_t}$

as the probability for the j^{th} word to appear knowing that the text is spam, as specified in the description.

5. $\mathbf{P}(Spam \mid [W_j = \text{true}]) = \frac{\mathbf{P}(Spam) \times \mathbf{P}([W_j = \text{true}] \mid Spam)}{\sum_{Spam} \mathbf{P}(Spam) \times \mathbf{P}([W_j = \text{true}] \mid Spam)}$

the probability for the email to be spam knowing that the j^{th} word appears in the text.

6. $\mathbf{P}(Spam \mid W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}) = \frac{\mathbf{P}(Spam) \times \prod_{i=0}^{N-1} \mathbf{P}(W_i \mid Spam)}{\sum_{Spam} \mathbf{P}(Spam) \times \prod_{i=0}^{N-1} \mathbf{P}(W_i \mid Spam)}$

The denominator appears to be a normalization constant. To eliminate it when deciding if we are dealing with spam, an easy trick is to compute the ratio:

$$\frac{\mathbf{P}([Spam = \text{true}] \mid W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}{\mathbf{P}([Spam = \text{false}] \mid W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})}$$

$$= \frac{\mathbf{P}([Spam = \text{true}]) \times \prod_{i=0}^{N-1} \mathbf{P}(W_i \mid [Spam = \text{true}])}{\mathbf{P}([Spam = \text{false}]) \times \prod_{i=0}^{N-1} \mathbf{P}(W_i \mid [Spam = \text{false}])}$$

This computation is faster and easier because requires only 2N products.

2.16 Bayesian program = Description + Question

Finally, a Bayesian program will always have the following simple structure:

$$\text{Program}(\mathbf{Pr})\begin{cases}\text{Description}(\mathbf{Ds})\begin{cases}\text{Specification}(\mathbf{Sp})\begin{cases}\text{Variables}(\mathbf{Va})\\ \text{Decomposition}(\mathbf{Dc})\\ \text{Forms}(\mathbf{Fo})\end{cases}\\ \text{Identification}(\mathbf{Id})\end{cases}\\ \text{Question}(\mathbf{Qu})\end{cases} \tag{2.18}$$

The Bayesian spam filter program is completely defined by:

$$
\mathbf{Pr}\begin{cases}
\mathbf{Ds}\begin{cases}
\mathbf{Sp}\begin{cases}
\mathbf{Va} \rightarrow Spam, W_0, W_2, ..., W_{N-1} \\
\mathbf{Dc} \rightarrow \begin{cases}
\mathbf{P}(Spam \wedge W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1}) \\
= \mathbf{P}(Spam) \times \prod_{i=1}^{N} \mathbf{P}(W_i \mid Spam)
\end{cases} \\
\mathbf{Fo} \rightarrow \begin{cases}
\mathbf{P}(Spam) \rightarrow \begin{cases}
\mathbf{P}([Spam = \text{false}]) = 0{,}25 \\
\mathbf{P}([Spam = \text{true}]) = 0{,}75
\end{cases} \\
\mathbf{P}(W_i \mid Spam) \rightarrow \begin{cases}
\mathbf{P}([W_i = \text{false}] \mid [Spam = \text{false}]) = 1 - \frac{1 + n_f^i}{2 + n_f} \\
\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{false}]) = \frac{1 + n_f^i}{2 + n_f} \\
\mathbf{P}([W_i = \text{false}] \mid [Spam = \text{true}]) = 1 - \frac{1 + n_t^i}{2 + n_t} \\
\mathbf{P}([W_i = \text{true}] \mid [Spam = \text{true}]) = \frac{1 + n_t^i}{2 + n_t}
\end{cases}
\end{cases}
\end{cases} \\
\mathbf{Id} \rightarrow \text{Parameters learned from instances}
\end{cases} \\
\mathbf{Qu} \rightarrow \mathbf{P}(Spam \mid W_0 \wedge ... \wedge W_j \wedge ... \wedge W_{N-1})
\end{cases}
\tag{2.19}
$$

2.17 Results

i	Word i	n_f^i	n_t^i
0	fortune	0	375
1	next	125	0
2	programming	250	0
3	money	0	750
4	you	125	375

Table 2.1: Hiistogram derived from an analysis of 1000 emails. The values n_f^i and n_t^i denote the number of emails that contained the i^{th} word in nonspam and spam emails respectively.

If we consider a spam filter with an N word dictionary, then any given email contains one and only one of the 2^N possible subsets of the dictionary. Here we restrict our spam filter

to a five word dictionary so that we can analyze the $2^5 = 32$ subsets.

Assume that a set of 1000 emails is used in the identification phase and that the resulting numbers of nonspams and spam emails are 250 and 750 respectively. Assume also that the resulting histogram tables for n_f^i and n_t^i are those shown in Table 2.1. The

i	$\mathbf{P}(W_i \mid Spam = \text{false})$		$\mathbf{P}(W_i \mid Spam = \text{true})$	
	$W_i = \text{false}$	$W_i = \text{true}$	$W_i = \text{false}$	$W_i = \text{true}$
0	**0.996032**	**0.00396825**	**0.5**	**0.5**
1	**0.5**	**0.5**	**0.99867**	**0.00132979**
2	**0.00396825**	**0.996032**	**0.99867**	**0.00132979**
3	**0.996032**	**0.00396825**	**0.00132979**	**0.99867**
4	**0.5**	**0.5**	**0.5**	**0.5**

Table 2.2: The resulting distribution $\mathbf{P}(W_i \mid Spam)$ from Table 2.1.

computed distribution $\mathbf{P}(W_i \mid Spam)$ is given in Table 2.2. We can now give all the possibilities of classification for an email and the probability of its being spam (Table 2.3).

Subset	W_0 fortune	W_1 next	W_2 programming	W_3 money	W_4 you	$\mathbf{P}(Spam \mid W_0 \wedge W_1 \wedge W_2 \wedge W_3 \wedge W_4)$	
						$Spam = \text{false}$	$Spam = \text{true}$
1	false	false	false	false	false	0.497351	0.502649
2	false	false	false	false	true	0.497351	0.502649
3	false	false	false	true	false	5.24907e-06	0.999995
4	false	false	false	true	true	5.24907e-06	0.999995
5	false	false	true	false	false	0.999995	5.36149e-06
6	false	false	true	false	true	0.999995	5.36149e-06
7	false	false	true	true	false	0.497351	0.502649
8	false	false	true	true	true	0.497351	0.502649
9	false	true	false	false	false	0.998656	0.00134393
10	false	true	false	false	true	0.998656	0.00134393
11	false	true	false	true	false	0.00392659	0.996073
12	false	true	false	true	true	0.00392659	0.996073
13	false	true	true	false	false	0.99999999	7.13918e-09
14	false	true	true	false	true	0.99999999	7.13918e-09
15	false	true	true	true	false	0.998656	0.00134393
16	false	true	true	true	true	0.998656	0.00134393

Table 2.3: The distribution $\mathbf{P}(Spam \mid W_0 \wedge W_1 \wedge W_2 \wedge W_3 \wedge W_4)$.

Subset	W_0 fortune	W_1 next	W_2 programming	W_3 money	W_4 you	$\mathbf{P}(Spam \mid W_0 \wedge W_1 \wedge W_2 \wedge W_3 \wedge W_4)$	
						$Spam$ = false	$Spam$ = true
17	true	false	false	false	false	0.00392659	0.996073
18	true	false	false	false	true	0.00392659	0.996073
19	true	false	false	true	false	2.09127e-08	0.99999999
20	true	false	false	true	true	2.09127e-08	0.99999999
21	true	false	true	false	false	0.998656	0.00134393
22	true	false	true	false	true	0.998656	0.00134393
23	true	false	true	true	false	0.00392659	0.996073
24	true	false	true	true	true	0.00392659	0.996073
25	true	true	false	false	false	0.747506	0.252494
26	true	true	false	false	true	0.747506	0.252494
27	true	true	false	true	false	1.57052e-05	0.999984
28	true	true	false	true	true	1.57052e-05	0.999984
29	true	true	true	false	false	0.999998	1.79193e-06
30	true	true	true	false	true	0.999998	1.79193e-06
31	true	true	true	true	false	0.747506	0.252494
32	true	true	true	true	true	0.747506	0.252494

Table 2.3: The distribution $\mathbf{P}(Spam \mid W_0 \wedge W_1 \wedge W_2 \wedge W_3 \wedge W_4)$.

Observe the distribution $\mathbf{P}(W_4 \mid Spam)$ on Table 2.2, the word, "you", provides no information at all, because it contributes of two uniform distributions. Consequently, the value of W_4 does not change anything in $\mathbf{P}(Spam \mid W_0 \wedge W_1 \wedge W_2 \wedge W_3 \wedge W_4)$. Observe the probability values of subsets 1-2, 3-4, 5-6, etc.

In effect, a single word can provide much, little or no information when classifying an email. For example, consider subset number 11 ({next, money}). Subsets 3, 12, 15, and 27 are within a distance 1 from this set. Adding or erasing a single word from set 11 has

different effects on the spam probability computation (Table 2.4).

Subset number	Subset	$\mathbf{P}(Spam \mid W_1 \wedge W_2 \wedge W_3 \wedge W_4 \wedge W_5)$	
		$Spam$ = false	$Spam$ = true
3	{money}	5.24907e-06	0.999995
11	{next, money}	0.00392659	0.996073
12	{next, money, you}	0.00392659	0.996073
15	{next,programming,money}	0.998656	0.00134393
27	{fortune,next,money}	1.57052e-05	0.999984

Table 2.4: Adding or subtracting a single word from subset 11 can change the probability of an email being spam gretly (set 15), a little (sets 3 and 27), or not at all (set 12).

3

Incompleteness and Uncertainty

A very small cause which escapes our notice determines a considerable effect that we cannot fail to see, and then we say that the effect is due to chance.

H. Poincare $$$

The goal of this chapter is twofold: (i) to present the concept of *incompleteness* and (ii) to demonstrate how incompleteness is the source of *uncertainty.* Two experiments explore these two topics. The first one is a very simple physical experiment called the "beam in the bin", the second one is a model for a water treatment unit.

3.1 The "beam in the bin" investigation

In this section, we present the "beam in the bin" experiment. This experiment is based on the simplest sensory-motor device interacting with the simplest real physical environ-

ment we could imagine. The goal of this extreme simplicity is to demonstrate that even in these conditions it is impossible to build a complete model of such an interaction: there is no way to get rid of hidden variables. Hidden variables are ignored by the model and consequently some (or all) model predictions may not be in accord with the real results. A "real" (*i.e.* not formal) phenomenon and its model never have the exact same behavior because of incompleteness.

In the following we describe the "beam in the bin" experimental set-up (Figure 3.1.).

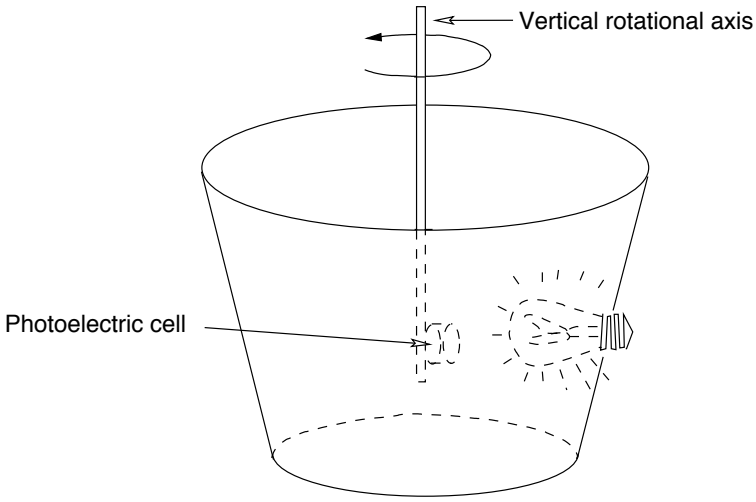

Figure 3.1: The "beam in the bin" experimental set-up. A photoelectric cell explores 72 different positions around a vertical axis measuring the reflected beam of a lamp. Both the cell and the light are placed inside a bin.

The environment consists of a green plastic bin in which a halogen lamp is fixed. The lamp light up the inside of the bin, where multiple reflections occur.

The sensory-motor device consists of a photoelectric cell suspended from a vertical metal bar that can turn around the vertical axis. Consequently, the photoelectric cell can rotate through 360° and we can control its position θ in increments of 5°.

We can also read the output of the photoelectric cell ρ, which has discrete values ρ in the range from 0 (no light) to 2047 (saturation).

The experimental protocol is as follows: we rotate the photoelectric cell, exploring 72 different positions (360 degrees by 5 degree increments) and collect the corresponding light reading for each position. Each position is measured 100 times. Using these 7200 measures, we build the histogram of the reading presented in Figure 3.2 and Figure 3.3. For each possible position and each possible reading, the number of times it has been observed is recorded.

Figure 3.2: Histogram of the light readings ρ according to angular orientation θ.

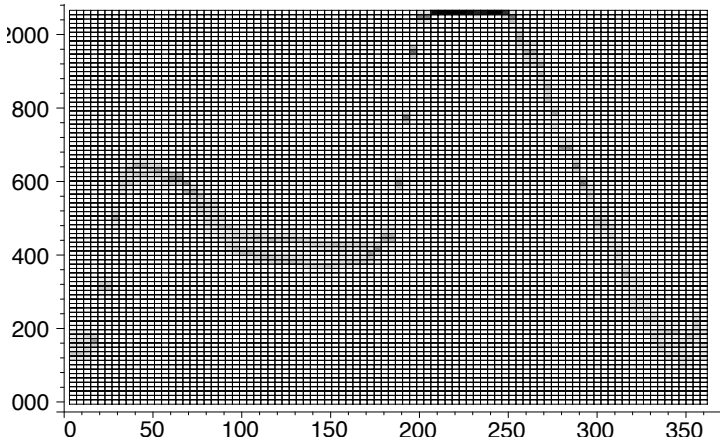

Figure 3.3: The density plot of the histogram on Figure 3.2. Light colors show less frequent values.

This histogram has a quite obvious interpretation. The main peak corresponds to the angular position where the cell is facing the light (angular position of 210°). The secondary peak corresponds to the opposite direction (30°) where the cell is facing the main reflection.

However, if we pay more attention to this data then some curious features appear.

Starting facing the light, turning left or right should not make any difference. The setup has a symmetry of revolution around the vertical axis and there is apparently nothing to break it. Consequently, the measured values of ρ should be symmetrical around the axis of lighting ($\theta = 210°$), but this is obviously not the case.

Trying to explain this absence of symmetry, the first idea is to check the position and inclination of the vertical axis. If it were not perfectly centered and vertical, the symmetry would be broken. Careful verifications of both position and inclination proved that they are both correct.

The next idea that comes to mind is that there may be other sources of light (for

example lamps, neons, or windows) around the bin that break the symmetry. This possibility is rather unlikely, as the halogen lamp is very powerful and very close to the photoelectric cell. Nevertheless, we check for this hypothesis by repeating the experiment with a black curtain over the bin. The asymmetry is still there.

Pursuing the investigation, we then check for dust on the inside wall of the bin. The bin is clean and glossy.

Much more complicated, we then check the quality of the angular control. Indeed, the vertical axis is controlled by an industrial robotic arm connected to a real-time control processor, itself the slave of a workstation. A complicated setup is required in order to use only one degree of freedom: the rotation of the robot's wrist. We arrange a complete new experiment to verify that when ordered to go to a given θ, the correct angular position is actually reached. We then discover a bug in one of the low-level control programs of the wrist. Some of the angles are not reached with the required accuracy. This robot had been used for years without anyone noticing this bug. However, this is not yet sufficient to explain the asymmetry of the data.

Finally, we suspect the photoelectric cell. Once again, we arrange a new experiment to test the cell. We then discover that it suffers ome hysteresis: the reading at time t is not independent of the previous ones. When the cell comes from a brighter zone, the reading is biased toward higher values. When collecting the data, to spend less time, we explore the θ values in sequence, going from 0° to 360°. These two facts explain the asymetry at least partly: after each bright area there is a plateau of high readings because of the hysteresis of the cell.

However, that is not completely the end of the story. By looking even more carefully at the data you may notice that it seems that in some areas there are two peaks rather than a single one (see Figure 3.4). We cannot find the exact reason for this but we remember that this data was collected in two runs on two different days. Certainly, some conditions such as temperature, humidity, and noice changed between the two runs and this may explain the sligt shift. Again, a "real" phenomenon and its model never have the exact same behavior because of incompleteness.

3.2 Observing a water treatment unit

The uncertainty of a phenomenon has two causes: (i) inaccuracies in the model and (ii) ignored variables. In this section, we demostrate this fact by a second experiment: the

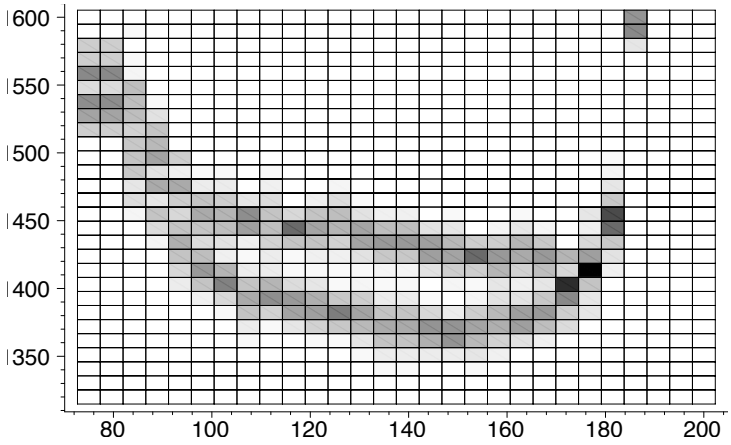

Figure 3.4: A zoom on the reading and orientation density plot presented on Figure 3.3 reveals that in some areas two peaks exist for the same orientation. Light colors are less frequent values.

water treatment unit. This experiment consists of two stages. In the first stage, we describe the *complete* model of the water treatment unit, giving all the variables and functions involved in the model. In the second stage, we pretend that some of the variables and functions of the model are not disponible. In other words, we generate a synthetic incompleteness of our model. The goal is to show the consequences of this incompleteness and to present a first step towards Bayesian modeling.

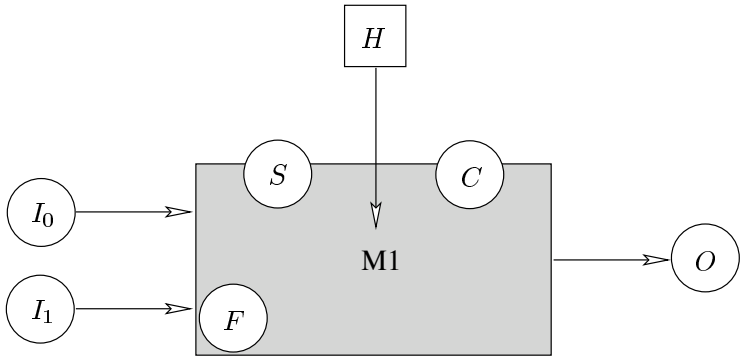

Figure 3.5: The treatment unit receives two water streams of quality I_0 and I_1 and generates an output stream of quality O. The resulting quality depends on I_0, I_1, two unknown variables H and F, and a control variable C. An operator regulates C, while the value of F is estimated by a sensor variable S.

3.2.1 The elementary water treatment unit

We now describe the complete model of the water treatment unit. Figure 3.5 is a schematic representation. The unit takes two water streams as inputs with respective water

qualities I_0 and I_1. Two different streams are used because partly purified water is recycled to dilute the more polluted stream, to facilitate its decontamination.

The unit produces an output stream of quality O.

The internal functioning state of the water treatment unit is described by the variable F. This variable F quantifies the efficiency of the unit but is not directly measurable. For instance, as the sandboxes becomes more loaded with contaminants the purification becomes less and less efficient and the value of F becomes lower and lower.

A sensor S helps to estimate the efficiency F of the unit.

A controller C is used to regulate and optimize O, the quality of the water in the output stream.

Finally, some external factor H may disturb the operation of the unit. For instance, this external factor could be the temperature or humidity of the air.

For didactic purposes, we consider that these seven variables may each take 11 different integer values ranging from 0 to 10. The value 0 is the worst value for I_0, I_1, F, and O, and 10 is the best.

When all variables have their nominal values, the ideal quality Q of the output stream is given by the equation:

$$Q = \textbf{INT}\left(\frac{I_0 + I_1 + F}{3}\right) \tag{3.1}$$

Where $\textbf{INT}(x)$ is the integer part of x.

The value of Q never exceeds the value O^*, reached when the unit is in perfect condition, with

$$O^* = \textbf{INT}\left(\frac{I_0 + I_1 + 10}{3}\right) \tag{3.2}$$

The external factor H may reduce the ideal quality Q and the control C may try to compensate for this disturbance or the bad condition of the treatment unit because of F. Consequently, the output quality O is obtained according to the following equations:

$$\alpha = \mathbf{INT}\left(\frac{I_0 + I_1 + F + C - H}{3}\right) \tag{3.3}$$

$$O = \begin{cases} \alpha & \text{if } (0 \le \alpha \le O^*) \\ (2O^* - \alpha) & \text{if } (\alpha > O^*) \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

We consider the example of a unit directly connected to the sewer: $[I_0 = 2], [I_1 = 8]$.

When $[C = 0]$ (no control) and $[H = 0]$ (no disturbance), Figure 3.6 gives the value of the quality O according to F, $(O^* = 6)$.

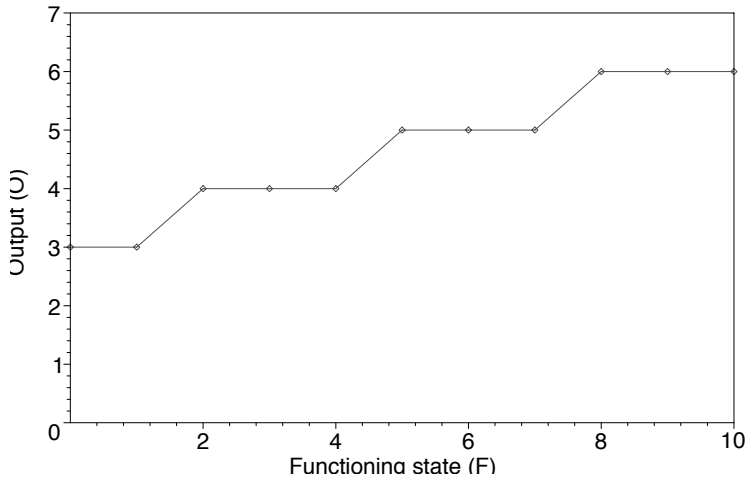

Figure 3.6: The output O as a function of the functioning state F with inputs, control and external factor fixed to: $[I_0 = 2] \wedge [I_1 = 8] \wedge [C = 0] \wedge [H = 0]$.

When the state of operation is not optimal (F different from 10), it is possible to compensate using C. However, if we over-control, then it may happen that the output deteriorates. For instance, if $[I_0 = 2], [I_1 = 8], [F = 8], [H = 0]$, the outputs obtained for the different values of C are shown in Figure 3.7.

The operation of the unit may be degraded by H. For instance, if $[I_0 = 2], [I_1 = 8], [F = 8], [C = 0]$, the output obtained for the different values of H are shown in Figure 3.8.

Finally, the value of the sensor S depends on I_0 and F as follows:

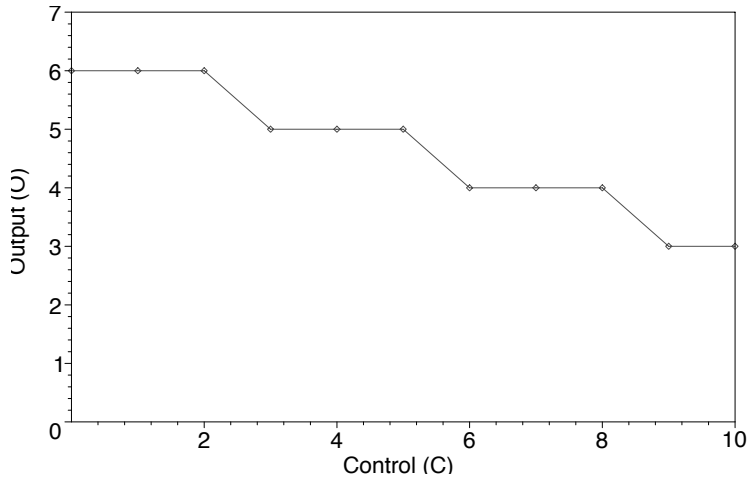

Figure 3.7: The output O as a function of control C with inputs, functioning state and external factor fixed to: $[I_0 = 2], [I_1 = 8], [F = 8], [H = 0]$.

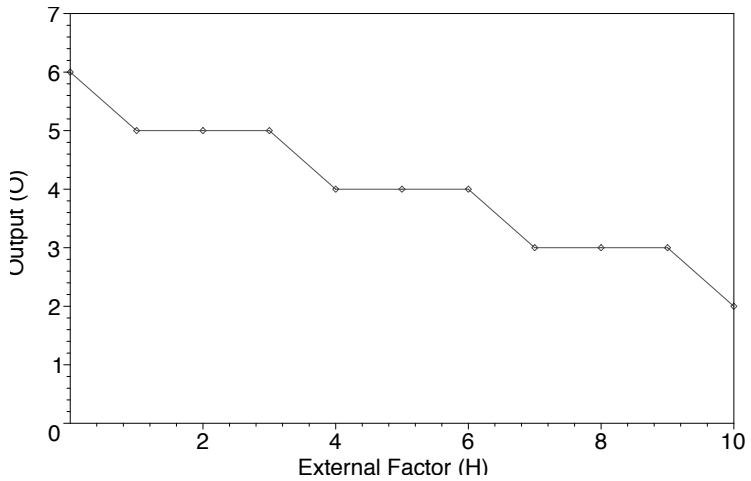

Figure 3.8: The output O as a function of the external factor H with inputs, functioning state and control fixed to: $[I_0 = 2], [I_1 = 8], [F = 8], [C = 0]$.

$$S = \textbf{INT}\left(\frac{I_0 + F}{2}\right) \tag{3.5}$$

The outputs of S in the 121 possible situations for I_0 and F are shown in Figure 3.9. Note that, if we know I_0, I_1, F, H, and C, we know with certainty the values of both S and O. At this stage, our water treatment unit is a completely deterministic process. Consequently, a complete model can be constructed.

Now consider what happens if we ignore the exact equations that rule the water treat-

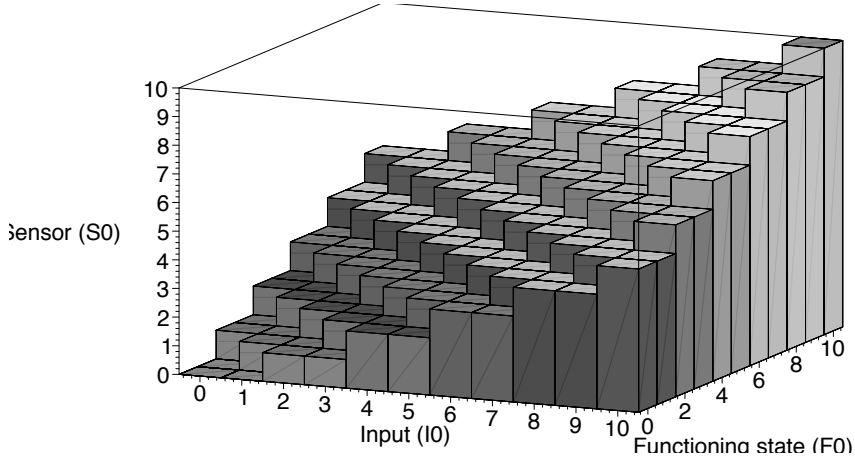

Figure 3.9: The sensor S as a function of input I_0 and functioning state F.

ment unit and, of course, the existence of the external factor H. The starting point for constructing our own model is limited to knowing the existence of the variables I_0, I_1, F, S, C, and O, and that the value of O on $I_0 \wedge I_1 \wedge S \wedge C$ and that of S depends on $I_0 \wedge F$.

What do we need to do now? Observe the behavior of the water treatment unit, in particular the quality of the output stream O, for different values of $I_0 \wedge I_1 \wedge S \wedge C$, as well as the sensor value S for different values of I_0 (remember that F cannot be observed). During these observations you will note that there are different situations in which uncertainty appears. The goal of the following section is to discuss this uncertainty.

3.2.2 Experimentation and uncertainty

Uncertainty on O because of inaccuracy of the sensor S

A given value of S corresponds to several possible values of I_0 and F. For instance, seven pairs of values of $I_0 \wedge F$ correspond to $[S = 1]$ in Figure 3.9.

Worse tha this, even knowing I_0 and S, two values of F are possible most of the time (see Figure 3.10). This fact will introduce some noise in the prediction of O.

To illustrate this effect let us first experiment with $H = 0$: the operation of the water treatment unit is not disturbed. For $[I_0 = 2], [I_1 = 8], [C = 2]$, we can explore the different possible values of the output O when F varies. However, as F is not directly observable, we can only collect data concerning S and O. These data are presented on Figure 3.11.

For some S it is possible to predict exactly the output O:

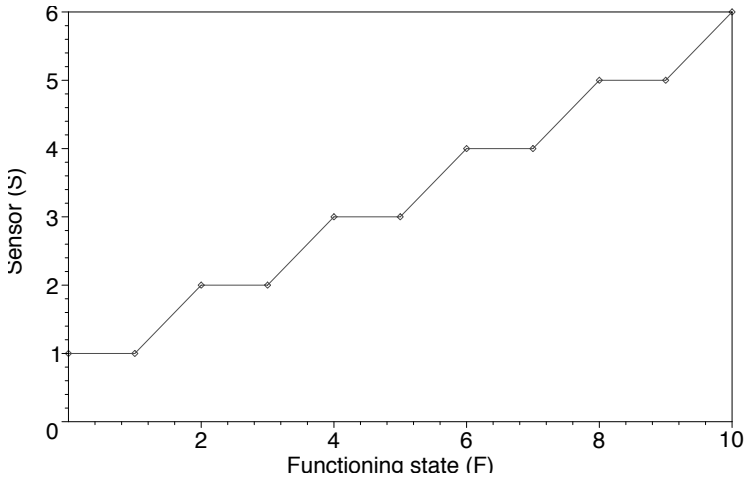

Figure 3.10: The sensor reading S as a function of the functioning sate F when the input $[I_0 = 2]$.

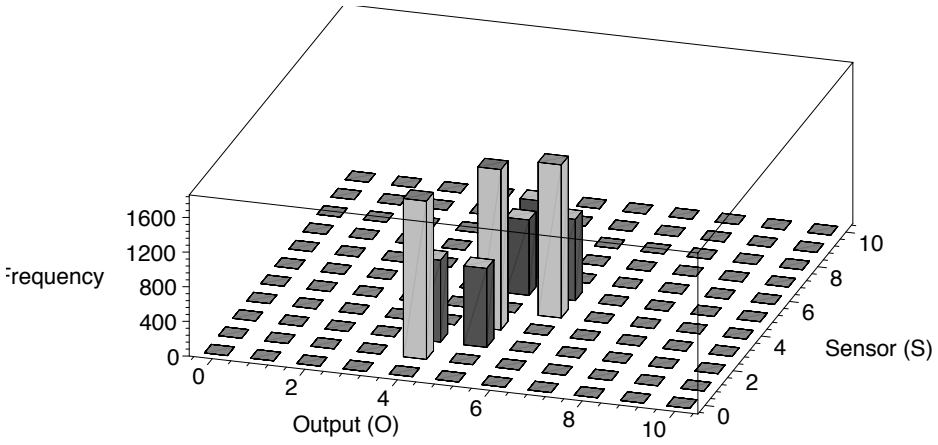

Figure 3.11: Histogram of the observed sensor state S and the output O when the inputs, the control and the external factor are fixed to $[I_0 = 2], [I_1 = 8], [C = 2], [H = 0]$ and the internal function F is generated randomly with a uniform distribution.

1. $[S = 1] \Rightarrow [O = 4]$

2. $[S = 3] \Rightarrow [O = 5]$

3. $[S = 4] \Rightarrow [O = 6]$

4. $[S = 6] \Rightarrow [O = 5]$

For some other values of S it is not possible to predict the potput O with certainty:

1. If $[S = 2]$, then O may take the value either four or five, with a slightly higher probability for four. Indeed, when $[S = 2]$, then F may be either two or three (see Figure 3.10) and, O will, respectively, be either four or five.

2. If $[S = 5]$, then O may take the value either five or six, with a slightly lower probability for five. When $[S = 5]$, F may be either eight or nine.

Some uncertainty appears here because of the inaccuracy of the sensor S, which may return the same reading for two different situations.

Uncertainty because of the ignored variable H

Let us now do the same experiment for a disturbed process (value of H drawn at random from the 11 possible values). Of course, we obtain different results with more uncertainty due to the effect on the output of the hidden variable H. The obtained data when $[I_0 = 2] \wedge [I_1 = 8] \wedge [C = 2]$ is presented on Figure 3.12.

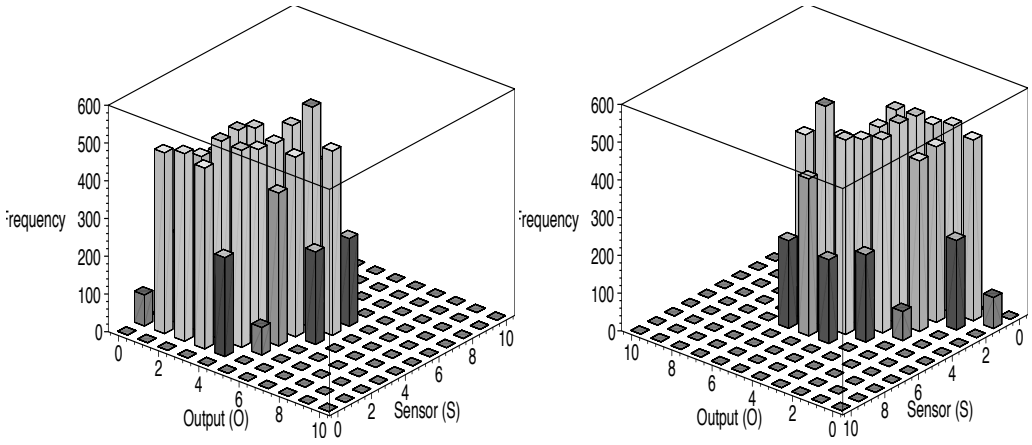

Figure 3.12: Two views of the histogram of the observed sensor state S and the output O when the inputs and the control are set to $[I_0 = 2], [I_1 = 8], [C = 2]$ and the values of the external factor and the internal functioning $H \wedge F$ are generated randomly with an uniform distribution.

In contrast with our previous experiment, this time no the value of S is sufficient to infer the value of O exactly.

The dispersion of the observations is the direct translation of the effect of H. Taking into account the effect of hidden variables such as H and even measuring their importance is one of the major challenges that Bayesian Programming must face. This is not an easy task when you are not even aware of the nature and number of these hidden variables!

3.3 Lessons, comments and notes

3.3.1 The effect of incompleteness

We assume that any model of a "real" (*i.e.* not formal) phenomenon is *incomplete*. There are always some hidden variables, not taken into account in the model, that influence the phenomenon. Furthermore, this incompleteness is irreducible: for any physical phenomenon, there is no way to build an exact model with no hidden variables[1].

The effect of these hidden variables is that the model and the phenomenon never have exactly reproducible behavior. *Uncertainty* appears as a direct consequence of this *incompleteness*. Indeed, the model may not completely take into account the data and may not predict exactly the behavior of the phenomenon[2].

For instance, in the above example, the influence of the hidden variable H, makes it impossible to predict with certainty the output O given the inputs I_0 and I_1, the reading of the sensor S, and the control C.

3.3.2 The effect of inaccuracy

The above example also demonstrates that there is another source of uncertainty: the *inaccuracy* of the sensors.

By inaccuracy, we mean that a given sensor may read the same value for different underlying situations. Here, the same reading on S may correspond to different values of F.

F is not a hidden variable as it is taken into account by the model. However, F cannot be measured directly and exactly. The values of F can only be inferred indirectly through the sensor S and they cannot be inferred with certainty.

It may be seen as a weak version of *incompleteness*, where a variable is not completely

1. See *FAQ/FAM 16.16 "Incompleteness irreducibility", page 202* for further discussion of that matter.
2. See *FAQ/FAM 16.24 "Noise or ignorance?", page 203* for more information on this subject.

hidden but is only partially known and accessible. Even though it is weak, this incompleteness still generates uncertainty.

3.3.3 Not taking into account the effect of ignored variables may lead to wrong decisions

Once the effects of the irreducible incompleteness of models eare recognized, a programmer must deal with them either ignoring them and using the incomplete models, or trying to take incompleteness into account using a probabilistic model.

Using a probabilistic model clearly appears to be the better choice as it will always lead to better decisions, based on more information than the nonprobabilistic one.

For instance, a nonprobabilistic model of our production unit, not taking into account the variable H would be, for instance[1]:

$$\alpha = \textbf{INT}\left(\frac{I_0 + I_1 + F + C}{3}\right) \tag{3.6}$$

$$O = \begin{cases} \alpha & \text{if } (0 \le \alpha \le O^*) \\ (2O^* - \alpha) & \text{if } (\alpha > O^*) \\ 0 & \text{otherwise} \end{cases} \tag{3.7}$$

$$S = \textbf{INT}\left(\frac{I_0 + F}{2}\right) \tag{3.8}$$

It would lead to false predictions of the output O and, consequently, to wrong control decision on C to optimize this output.

For instance, scanning the 11 different possible values for C when $[I_0 = 2], [I_1 = 8], [F = 8]$ and consequently $[S = 5]$ the above model predicts that indifferently for $[C = 0]$, $[C = 1]$, and $[C = 2]$, O will take its optimal value: six (see Figure 3.7).

The observations depict a somewhat different and more complicated "reality" as shown in Figure 3.13.

---

1. Note the absence of H

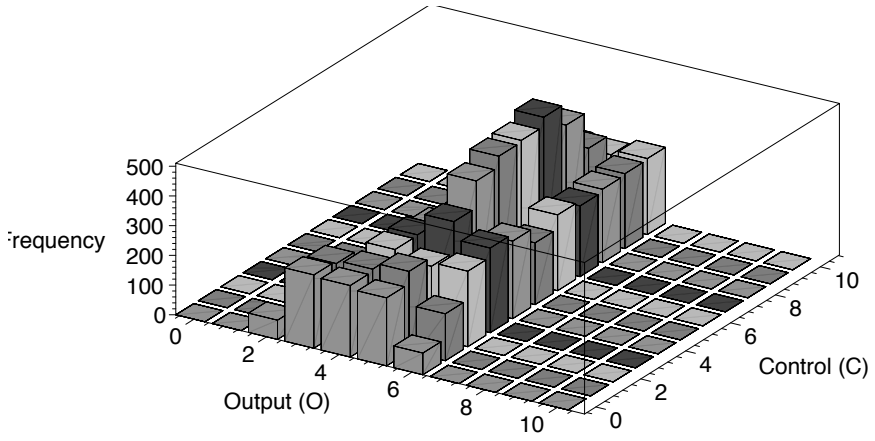

Figure 3.13: The output O as a function of the controller C when the inputs and the internal functioning are fixed to $[I_0 = 2], [I_1 = 8], [F = 8]$.

The choice of C to optimize O is now more complicated but also more informed.

The adequate choice of C to produce the optimal output $[O = 6]$ is now, with nearly equivalent probabilities, to select a value of C greater than or equal to two. Indeed, this is completely different choice from when the "exact" model is used!

3.3.4 From incompleteness to uncertainty

Any program that models a real phenomenon must face a central difficulty: how should it use an incomplete model of the phenomenon to reason, decide, and act efficiently?

Definition 3.1 *The purpose of Bayesian Programming is precisely to tackle this problem with a well-established formal theory: probability calculus. The sequel of this book will try to explain how to do this.*

In the Bayesian Programming approach, the programmer does not propose an exact model but rather expresses a probabilistic canvas in the specification phase. This probabilistic canvas gives some hints about what observations are expected. The specification is not a fixed and rigid model purporting completeness. Rather, it is a framework, with open parameters, waiting to be shaped by the experimental data. Learning is the means of setting these parameters. The resulting probabilistic *descriptions* come from both: (i) the views of the programmer and (ii) the physical specifics of each phenomenon. Even the influence of the hidden variables is taken into account and quantified; the more important their effects, the more noisy the data, and the more uncertain the resulting descriptions.

The theoretical foundations of Bayesian Programming may be summed up by Figure 3.14.

The first step in Figure 3.14 transforms the irreducible incompleteness into uncer-

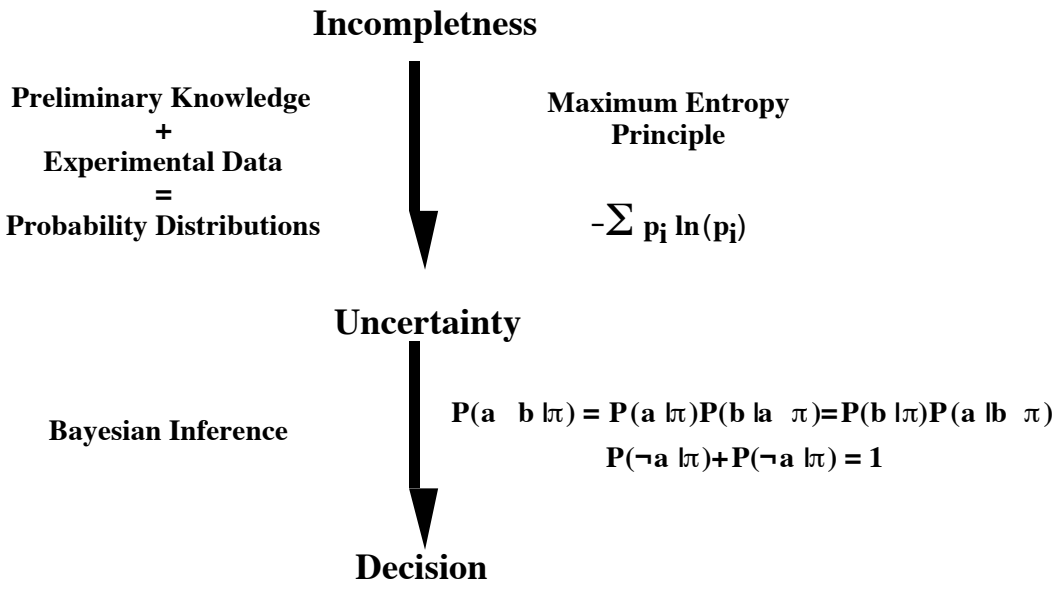

Figure 3.14: Theoretical foundation.

tainty. Starting from the specification and the experimental data, learning builds probability distributions.

The maximum entropy principle is the theoretical foundation of this first step. Given some specifications and some data, the probability distribution that maximizes the entropy is *the* distribution that *best* represents the combined specification and data. Entropy gives a precise, mathematical and quantifiable meaning to the quality of a distribution[1].

Two extreme examples may help to understand what occurs:

- Suppose that we are studying a formal phenomenon. There may not be any hidden variables. A complete model may be proposed. The phenomenon and the model could be identical. For instance, this would be the case if we take the equations of Section as the model of the phenomenon described in that same section. If we select this model as the specification, any data set will lead to a description made of Diracs. There is no uncertainty; any question may be answered either by true or

1. See *FAQ/FAM 16.22 "Maximum entropy principle justifications", page 203* for justifications for the use of the maximum entropy principle.

false. Logic appears as a special case of the Bayesian approach in that particular context (see Cox, 1979).

- At the opposite extreme, suppose that the specification consists of very poor hypotheses about the modeled phenomenon, for instance, by ignoring H and also the inputs I_0 and I_1 in a model of the above process. Learning will only lead to flat distributions, containing no information. No relevant decisions can be made, only completely random ones.

Hopefully, most common cases are somewhere between these two extremes. Specification, even imperfect and incomplete, is relevant and provides interesting hints about the observed phenomenon. The resulting descriptions are neither Diracs nor uniform distributions. They give no certitudes, although they provide a mean of taking the best possible decision according to the available information. This is the case here when the only hidden variable is H.

The second step in Figure 3.14 consists of reasoning with the probability distributions obtained by the first step.

To do so, we only require the two basic rules of Bayesian inference presented in Chapter 2. These two rules are to Bayesian inference what the resolution principle is to logical reasoning (see Robinson, 1965; Robinson, 1979; Robinson & Sibert, 1983a; Robinson & Sibert, 1983b). These inferences may be as complex and subtle as those usually achieved with logical inference tools, as will be demonstrated in the different examples presented in the sequel of this book.

4

Description = Specification + Identification

The scientific methodolgy forbids us to have an opinion on questions which we do not understand, on questions which we do not know how to put clearly. Before anything else, we must know how to state problems. In science problems do not appear spontaneously. This "sens of problem" is precisely what characterize a true scientific mind. For such a mind, any knowledge is an answer to a question. Without a question there cannot be scientific knowledge. Nothing is obvious. Nothing is given. Everything is built.[1]

Gaston Bachelard, La formation de l'esprit scientifique

---

1. L'esprit scientifique nous interdit d'avoir une opinion sur des questions que nous ne comprenons pas, sur des questions que nous ne savons pas poser clairement. Avant tout, il faut savoir poser les problemes. Et quoi qu'on dise, dans la vie scientifique, les problemes ne se posent d'eux-memes.C'est precisement ce "sens du probleme" qui donne la marque du veritable esprit scientifique. Pour un esprit scientifique, toute connaissance est une réponse a une question. S'il n'y a pas eu de question, il ne peut y avoir connaissance scientifique. Rien ne va de soi. Rien n'est donne. Tout est construit. (Bachelard, 1938)

In this chapter, we come back to the fundamental notion of *description*. A description is a probabilistic model of a given phenomenon. It is obtained after two phases of development:

1. A *specification* phase where the programmer expresses in probabilistic terms his own knowledge about the modelled phenomenon.

2. An *identification* phase where this starting probabilistic canvas is refined by learning from data.

Descriptions are the basic elements that are used, combined, composed, manipulated, computed, compiled, and questioned in different ways to build Bayesian programs.

4.1 Pushing objects and following contours

To introduce this notion of description we present two very simple robotic experiments where we want a small mobile robot named Khepera either to push objects or to follow their contours.

4.1.1 The Khepera robot

Khepera is a two-wheeled mobile robot, 57 millimeters in diameter and 29 milimeters in height, with a total weight of 80 grams (see Figure 4.1). It was designed at EPFL[1] and is commercialized by K-Team[2].

The robot is equipped with eight light sensors (six in front and two behind), which takes values between 0 and 511 in inverse relation to light intensity, stored in variables $L_0, ..., L_7$ (see Figure 4.2).

These eight sensors can also be used as infrared proximeters, taking values between 0 and 1023 in inverse relation to the distance from the obstacle, stored in variables $Px_0, ..., Px_7$ (see Figure 4.2).

The robot is controlled by the rotation speeds of its left and right wheels, stored in variables M_l and M_r, respectively.

From these 18 basic sensory and motor variables, we derive two new sensory variables (Dir and $Prox$) and one new motor variable (Rot). They are described below:

1. Ecole Polytechnique Fédérale de Lausanne (Switzerland)
2. http://www.K-team.com/

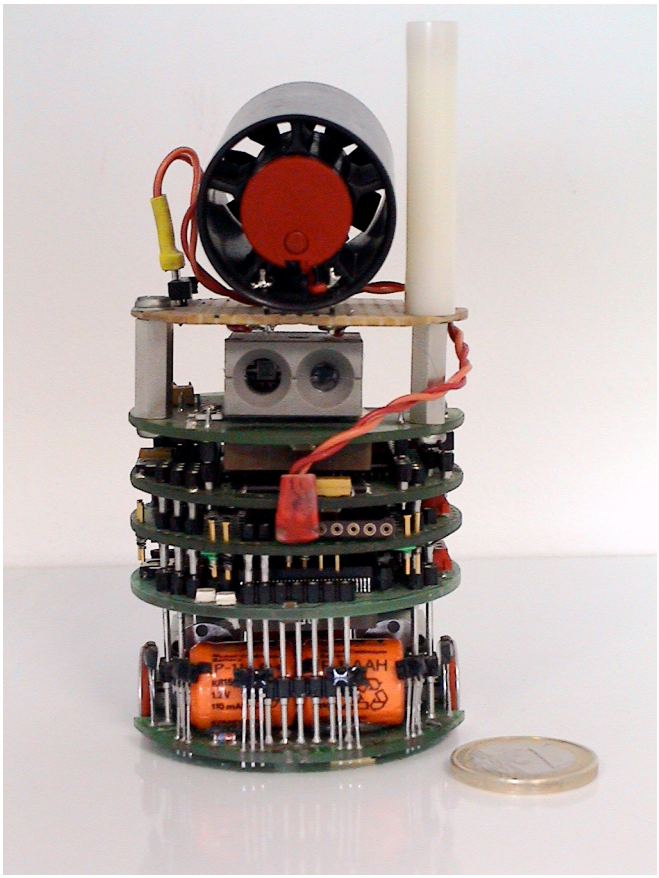

Figure 4.1: The Khepera mobile robot.

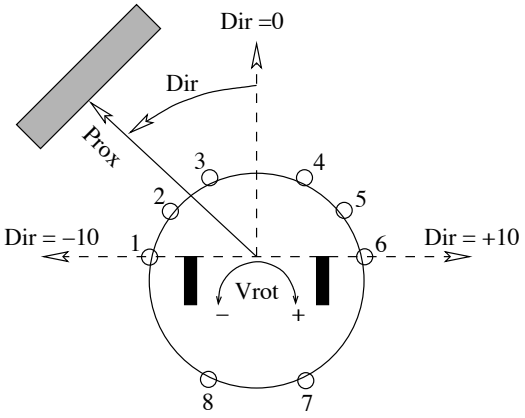

Figure 4.2: The sensory-motor variables of the Khepera robot.

- *Dir* is a variable that approximately corresponds to the bearing of the closest obstacle (see Figure 4.2). It takes values between -10 (obstacle to the left of the robot) and +10 (obstacle to the right of the robot), and is defined as follows:

$$Dir = \textbf{FLOOR}\left(\frac{90(Px_5 - Px_0) + 45(Px_4 - Px_1) + 5(Px_3 - Px_2)}{9(1 + Px_0 + Px_1 + Px_2 + Px_3 + Px_4 + Px_5)}\right) \quad (4.1)$$

- *Prox* is a variable that approximately corresponds to the proximity of the closest obstacle (see Figure 4.2). It takes values between 0 (obstacle a long way from the robot) and 15 (obstacle very close to the robot), and is defined as follows:

$$Prox = \textbf{FLOOR}\left(\frac{\textbf{MAX}(Px_0, Px_1, Px_2, Px_3, Px_4, Px_5)}{64}\right) \quad (4.2)$$

- The robot is piloted solely by its rotation speed (the translation speed is fixed). It receives motor commands from the *Rot* variable, calculated from the difference between the rotation speeds of the left and right wheels. *Rot* takes on values between -10 (fastest to the left) and +10 (fastest to the right).

4.1.2 Pushing objects

The goal of the first experiment is to teach the robot how to push objects.

First, in a *specification* phase, the programmer specifies his knowledge about this behavior in probabilistic terms.

Then, in a learning phase (*identification*), we drive the robot with a joystick to push objects. During that phase, the robot collects, every tenth of a second, both the values of its sensory variables and the values of its motor variables (determined by the joystick position). These data set are then used to identify the free parameters of the parametric forms.

Finally, in a restitution phase, the robot must reproduce the behavior it has just learned. Every tenth of a second it decides the values of its motor variables, knowing the values of its sensory variables and the internal representation of the task (the description).

Specification

Having defined our goal, we describe the three steps necessary to define the preliminary knowledge.

1. Chose the pertinent variables

2. Decompose the joint distribution

3. Define the parametric forms

Variables

First, the programmer specifies which variables are pertinent for the task.

To push objects it is necessary to have an idea of the position of the objects relative to the robot. The front proximeters provide this information. However, we chose to summarize the information from these six proximeters by the two variables Dir and $Prox$.

We also chose to set the translation speed to a constant and to operate the robot by its rotation speed Rot.

These three variables are all we require to push obstacles. Their definitions are summarized up as follows:

$$\begin{aligned} &Dir \in \{-10, ..., 10\}, \mathbf{Card}(Dir) = 21 \\ &Prox \in \{0, ..., 15\}, \mathbf{Card}(Prox) = 16 \\ &Rot \in \{-10, ..., 10\}, \mathbf{Card}(Rot) = 21 \end{aligned} \tag{4.3}$$

Decomposition

In the second specification step, we give a decomposition of the joint probability $\mathbf{P}(Dir \wedge Prox \wedge Rot)$ as a product of simpler terms.

$$\begin{aligned} &\mathbf{P}(Dir \wedge Prox \wedge Rot) \\ &= \mathbf{P}(Dir \wedge Prox) \times \mathbf{P}(Rot \mid Dir \wedge Prox) \end{aligned} \tag{4.4}$$

This equality simply results from the application of the conjunction rule (2.5).

Forms

To be able to compute the joint distribution, we must finally assign parametric forms to each of the terms appearing in the decomposition:

$$\begin{aligned} \mathbf{P}(Dir \wedge Prox) &\equiv \mathbf{Uniform} \\ \mathbf{P}(Rot \mid Dir \wedge Prox) &\equiv \mathbf{B}(\mu(Dir, Prox), \sigma(Dir, Prox)) \end{aligned} \tag{4.5}$$

We have no *a priori* information about the direction or distance of the obstacles.

Hence, $\mathbf{P}(Dir \wedge Prox)$ is a uniform distribution, with all directions and proximities having the same probability. As we have 21×16 different possible values for $Dir \wedge Prox$ we get:

$$\mathbf{P}(dir \wedge prox) = \frac{1}{21 \times 16} = \frac{1}{336} \tag{4.6}$$

For each sensory situation, we believe that there is one and only one rotation speed (Rot) that should be preferred. The distribution $\mathbf{P}(Rot \mid Dir \wedge Prox)$ is thus unimodal. However, depending of the situation, the decision to be made for Rot may be more or less certain. This is presumed by assigning a Bell-shaped[1] parametric form to $\mathbf{P}(Rot \mid Dir \wedge Prox)$. For each possible position of the object relative to the robot we have a bell-shaped distribution. Consequently, we have $21 \times 16 = 336$ bell-shaped distributions and we have $2 \times 21 \times 16 = 772$ free parameters: 336 means and 336 standard deviations.

Identification

To set the values of these free parameters we drive the robot with a joystick (see Movie 1[2]), and collect a set of data.

Every tenth of a second, we obtain the value of Dir and $Prox$ from the proximeters and the value of Rot from the joystick. Let us call the particular set of data corresponding to this experiment δ-$push$. A datum collected at time t is a triplet $(rot^t, dir^t, prox^t)$. During the 30 seconds of learning, 300 such triplets are recorded.

From the collection δ-$push$ of such data, it is very simple to compute the corresponding values of the free parameters. We first sort the data in 336 groups, each corresponding to a given position of the object and then compute the mean and standard deviation of Rot for each of these groups. There are only 300 triplets for 336 groups.

Moreover, these 300 triplets are concentrated around some particular positions often observed when pushing obstacles. Consequently, it may often happen that a given position never occurs and that no data is collected for this particular situation. In that case, we set the corresponding mean to zero and the standard deviation to 10. The bell-shaped distribution is then flat, close to a uniform distribution.

1. Bell-shaped distributions are distributions on discrete variables that have a Gaussian shape. They are noted with the **B** symbol and defined by their mean and standard deviation as regular Gaussian on continuous variables.
2. Movie 1 at http://bayesian-programming.org/book/movies

Figure 4.3 presents three of the 336 curves. The first one corresponds to an obstacle

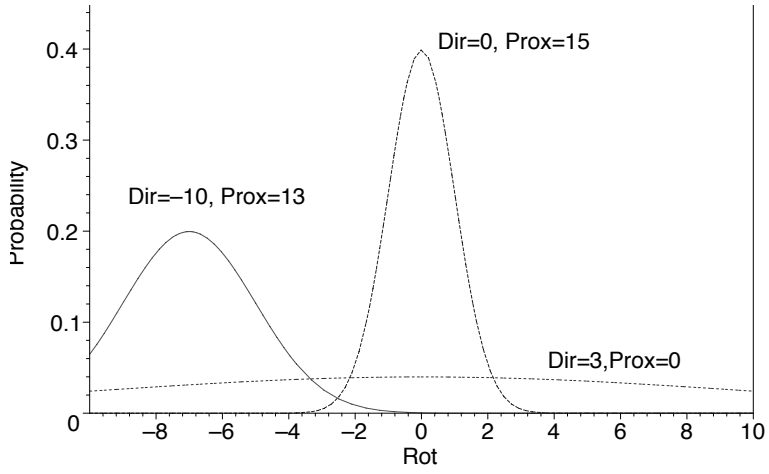

Figure 4.3: $\mathbf{P}(Rot \mid Dir \wedge Prox)$ when pushing objects for different situations.

very close to the left ($[Dir = -10], [Prox = 13]$), and shows that the robot should turn to the left rapidly with average uncertainty. The second one corresponds to an obstacle right in front and in contact ($[Dir = 0], [Prox = 15]$), and shows that the robot should go straight with very low uncertainty. Finally, the last one shows an unobserved situation where the uncertainty is maximal.

Results

To render the pushing obstacle behavior just learned, a decision on Rot is made every tenth of a second according to the following algorithm.

1. The sensors are read and the values of dir^t and $prox^t$ are computed

2. The corresponding distribution $\mathbf{P}(Rot \mid prox^t \wedge dir^t)$ is selected from the 336 distributions stored in memory.

3. A value rot^t is drawn at random according to this distribution and sent to the motors.

As shown in Movie 1[1], the Khepera learns how to push obstacles in 30 seconds. It learns the particular dependency, corresponding to this specific behavior, between the sensory variables Dir and $Prox$, and the motor variable Rot.

This dependency is largely independent of the particular characteristics of the objects (such as weight, color, balance, or nature). Therefore, as shown in Movie 2[2], the robot is also able to push different objects. This, of course, is only true within certain limits. For instance, the robot will not be able to push an object if it is too heavy.

4.1.3 Following contours

The goal of the second experiment is to teach the robot how to follow the contour of an object.

We will follow the same steps as in the previous experiment: first, a specification phase, then an identification phase where we also drive the robot with a joystick but this time to follow the contours.

We keep the exact same specification, changing only the data to be learned. The resulting description is however completely different: following contours of objects instead of pushing them.

Specification

Variables

To follow the contours, we must know where the object is situated relative to the robot. This is defined by the variables Dir and $Prox$, as in the previous experiment. We must also pilot the robot using its rotation speed with the variable Rot. The required variables are thus exactly the same as previously:

$$\begin{aligned} &Dir \in \{-10, \ldots, 10\}, \textbf{CARD}(Dir) = 21 \\ &Prox \in \{0, \ldots, 15\}, \textbf{CARD}(Prox) = 16 \\ &Rot \in \{-10, \ldots, 10\}, \textbf{CARD}(Rot) = 21 \end{aligned} \tag{4.7}$$

Decomposition

The decomposition does not change either:

1. Movie 1 at http://bayesian-programming.org/book/movies
2. Movie 2 at http://bayesian-programming.org/book/movies

$$\begin{aligned}&\mathbf{P}(Dir \wedge Prox \wedge Rot)\\ &= \mathbf{P}(Dir \wedge Prox) \times \mathbf{P}(Rot \mid Dir \wedge Prox)\end{aligned} \tag{4.8}$$

Forms

Finally, the parametric forms are also the same:

$$\begin{aligned}\mathbf{P}(Dir \wedge Prox) &\equiv \textsc{Uniform}\\ \mathbf{P}(Rot \mid Dir \wedge Prox) &\equiv \mathbf{B}(\mu(Prox, Dir), \sigma(Prox, Dir))\end{aligned} \tag{4.9}$$

We still have no idea of the possible position of the object relative to the robot and we still believe that for a given sensory situation, there is one, and only one, rotation speed that should be preferred.

Identification

In contrast, the learned data are completely different, because we are driving the robot to do some contour following (see Movie 3[1]). The learning process is the same but the data set, called *δ-follow*, is completely different.

The collection *δ-follow* of data leads to completely different values of the 336 means and standard deviation of the bell-shaped distributions. This clearly appears in the following distributions presented for the same relative positions of the object and the robot as in the previous experiment:

- Figure 4.4 shows the two distributions obtained after learning for both experiments (pushing objects and following contours) when the object is close to the left ($[Dir = -10], [Prox = 13]$). When pushing, the robot turns left to face the object; on the contrary, when following, the robot goes straight bordering the object.

- Figure 4.5 shows the two distributions obtained after learning for both experiments (pushing objects and following contours) when the object is in contact right in front of the robot ($[Dir = 0], [Prox = 15]$). When pushing, the robot goes straight. On the contrary, when following, the robot turns to the right to have the

1. Movie 3 at http://bayesian-programming.org/book/movies

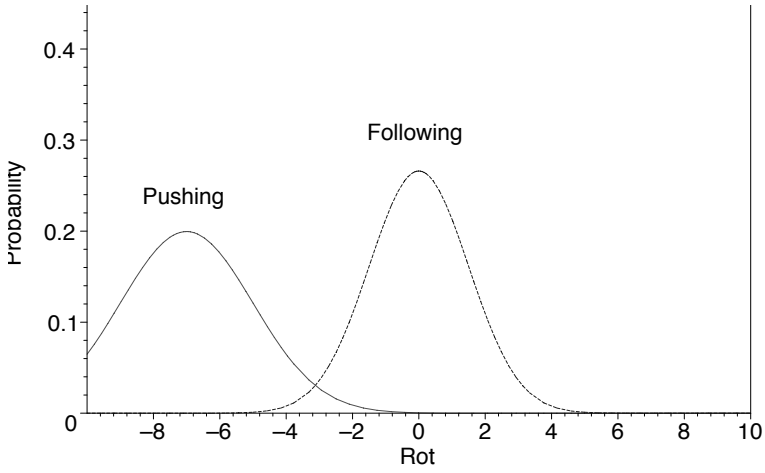

Figure 4.4: $\mathbf{P}(Rot \mid [Dir = -10] \wedge [Prox = 13])$
when pushing objects and when following contours.

object on its left. However, the uncertainty is large in this last case.

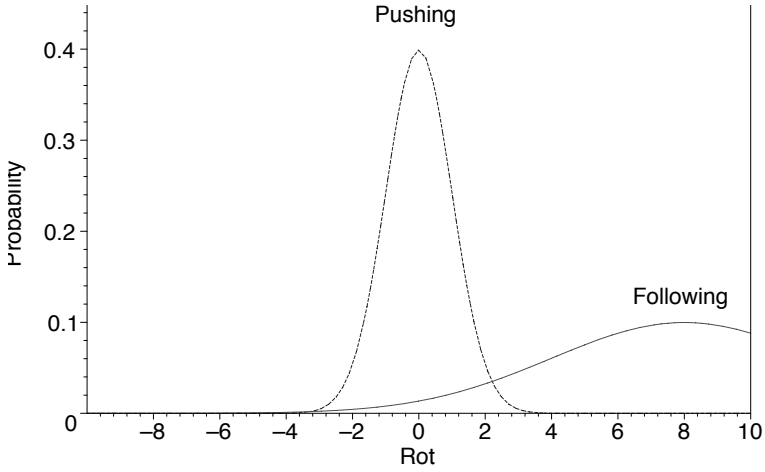

Figure 4.5: $\mathbf{P}(Rot \mid [Dir = 0] \wedge [Prox = 15])$
when pushing objects and when following contours.

Results

The restitution process is also the same, but as the bell-shaped distributiond are different,

the resulting behavior is completely different, as demonstrated by Movie 3.

It should be noted that one turn around the object is enough to learn the contour following behavior.

4.2 Description of a water treatment unit

Let us return to the previous example of a water treatment unit, and try to build the description of this process.

4.2.1 Specification

Variables

Following our Bayesian Programming methodology, the first step in building this description is to choose the pertinent variables.

The variables to be used by our Bayesian model are obviously the following:

$$I_0, I_1, F, S, C, O \in \{0, ..., 10\} \tag{4.10}$$

where every of this variables has a cardinality of 11. Of course H is missing.

Decomposition

Using the conjunction postulate (2.7) iteratively, we can write that the joint probability distribution on the six variables is equal to:

$$\begin{aligned} &\mathbf{P}(I_0 \wedge I_1 \wedge F \wedge S \wedge C \wedge O) \\ &= \mathbf{P}(I_0) \times \mathbf{P}(I_1 \mid I_0) \times \mathbf{P}(F \mid I_0 \wedge I_1) \times \mathbf{P}(S \mid I_0 \wedge I_1 \wedge F) \\ &\times \mathbf{P}(C \mid I_0 \wedge I_1 \wedge F \wedge S) \times \mathbf{P}(O \mid I_0 \wedge I_1 \wedge F \wedge S \wedge C) \end{aligned} \tag{4.11}$$

This is an exact mathematical expression.

The designer knows more about the process than this exact form. For instance, he or she knows that:

1. The qualities of the two input streams I_0 and I_1 are independent:

$$\mathbf{P}(I_0 \mid I_1) = \mathbf{P}(I_0) \tag{4.12}$$

2. The state of operation of the unit is independent of both entries:

$$\mathbf{P}(F \mid I_0 \wedge I_1) = \mathbf{P}(F) \tag{4.13}$$

3. The reading of the sensor depends only on I_0 and F. It does not depend on I_1.

$$\mathbf{P}(S \mid I_0 \wedge I_1 \wedge F) = \mathbf{P}(S \mid I_0 \wedge F) \tag{4.14}$$

4. The control C may not be established without knowing the desired output O. Consequently, as long as O is unknown, C is independent of the entries and of the state of operation.

$$\mathbf{P}(C \mid I_0 \wedge I_1 \wedge F \wedge S) = \mathbf{P}(C) \tag{4.15}$$

A few more thoughts may be necessary about this simplification, which is rather subtle. If you do know the desired output O, the control C will obviously depend on the entries, the state of operation and the reading of the sensor. However, if you do not know the objective, could you think of any reason to condition the control C on these variables? If you do not know where you want to go, do you have any good reason to choose a specific direction, even knowing the map and where you are?

5. The output O depends on I_0, I_1, F, S, and C. However, because of the presence of the sensor, there is some redundancy between I_0, F and S. If you know I_0 and F, then obviously knowing S does not bring any new information. This could be used to state: $\mathbf{P}(O \mid I_0 \wedge I_1 \wedge F \wedge S \wedge C) = \mathbf{P}(O \mid I_0 \wedge I_1 \wedge F \wedge C)$. Knowing I_0 and S, there is still some uncertainty about F (see Figures 3.9 & 3.10). However, as the value of F is not directly accessible for learning, we may consider as a first approximation that knowing I_0 and S, F may be neglected:

$$\mathbf{P}(O \mid I_0 \wedge I_1 \wedge F \wedge S \wedge C) = \mathbf{P}(O \mid I_0 \wedge I_1 \wedge S \wedge C) \tag{4.16}$$

Finally, the decomposition of the joint probability will be specified as:

$$\begin{aligned} &\mathbf{P}(I_0 \wedge I_1 \wedge F \wedge C \wedge S \wedge O) \\ &= \mathbf{P}(I_0) \times \mathbf{P}(I_1) \times \mathbf{P}(F) \times \mathbf{P}(S \mid I_0 \wedge F) \times \mathbf{P}(C) \times \mathbf{P}(O \mid I_0 \wedge I_1 \wedge S \wedge C) \end{aligned} \tag{4.17}$$

We see here a first example of the "art of decomposing" a joint distribution.

The decomposition is a means to compute the joint distribution and, consequently, answer all possible questions. This decomposition has the following qualities:

- It is a better model than the basic joint distribution, because we add some useful knowledge through points 1 to 5 above.
- It is easier to compute than the basic joint distribution, because instead of working in a six-dimensional space, we will do the calculation in spaces of smaller dimension (see Chapter 5 for more on this).
- It is easier (or at least possible) to identify. The simplification of point 5 has been made for that purpose.

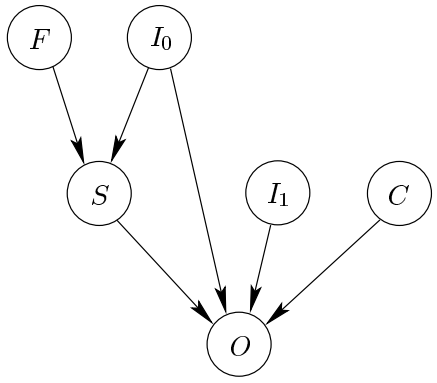

Figure 4.6: The graphical model of Expression (4.17), a water treatment unit.

Forms

To finish the specification task, we must still specify the parametric forms of the distribution appearing in the decomposition:

1. We have no a priori information on the entries I_0 and I_1:

$$\begin{aligned}\mathbf{P}(I_0) &\equiv \textbf{UNIFORM}\\ \mathbf{P}(I_1) &\equiv \textbf{UNIFORM}\end{aligned} \tag{4.18}$$

2. Neider do we have any a priori information on F:

$$\mathbf{P}(F) \equiv \textbf{UNIFORM} \tag{4.19}$$

3. We know an exact model of the sensor S:

$$\mathbf{P}(S \mid I_0 \wedge F) \equiv \delta_{\textbf{INT}\left(\frac{I_0+F}{2}\right)}(I_0 \wedge F) \tag{4.20}$$

Where:

$$\delta_{\textbf{INT}\left(\frac{I_0+F}{2}\right)} \tag{4.21}$$

is a Dirac distribution with probability one if and only if:

$$S = \textbf{INT}\left(\frac{I_0+F}{2}\right). \tag{4.22}$$

4. Not knowing the desired output O, all possible controls are equally probable:

$$\mathbf{P}(C) \equiv \textbf{UNIFORM} \tag{4.23}$$

5. Finally, each of the $11^4 = 11 \times 11 \times 11 \times 11$ distributions $\mathbf{P}(O \mid I_0 \wedge I_1 \wedge S \wedge C)$ (one for each possible value of $I_0 \wedge I_1 \wedge S \wedge C$) is defined as a histogram on the 11 possible values of O.

4.2.2 Identification

After the specification phase, we end up with 11^4 distributions with 11 free parameters each. Thus we have $11^5 = 161051$ free parameters to identify.

To do this we will run the simulator described in Chapter 3, drawing at random with

uniform distributions I_0, I_1, F, H and C. For each of these draws (for instance, 10^7 of them), we compute the corresponding values of the sensor S and the output O. We then update the 11^4 histograms according to the values of I_0, I_1, S, C, and O.

4.2.3 Results

Such histograms have already been presented in previous chapters, for instance, in Figure 3.12 reproduced below as Figure 4.7 may be seen as a collection of 11 of these histograms

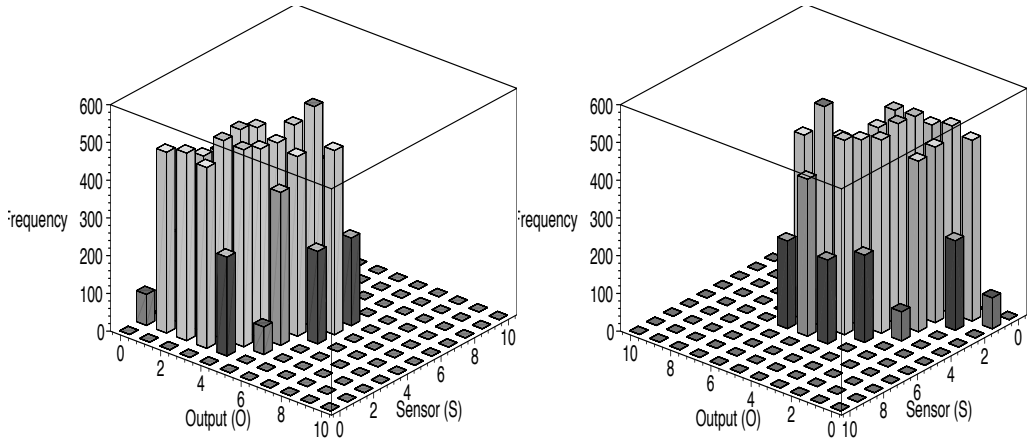

Figure 4.7: Two views of the collection of distributions represented by $\mathbf{P}(O \mid [I_0 = 2] \wedge [I_1 = 8] \wedge S \wedge [C = 2])$. The 11 values of S (0 to 10) generate the same number of distributions.

$\mathbf{P}(O \mid I_0 \wedge I_1 \wedge S \wedge C)$ when $[I_0 = 2], [I_1 = 8], [C = 2]$ and S varies. Note that now probability values are indicated rather than the number of observations. Consequently, the values have been normalized along each line. This operation changes the relative dimension of these lines.

The complete description of the elementary water treatment unit is made of 11^4 histograms, 11^3 times as much data as in Figure 4.7.

4.3 Lessons, comments and notes

4.3.1 Description = Specification + Identification

Descriptions are the basic elements that are used, combined, composed, manipulated, computed, compiled, and questioned in different ways to build Bayesian programs.

A description is the probabilistic model of the observed phenomenon.

As such, it results from both the a priori knowledge of the programmer about this phenomenon[1] and from the experimental data.

The programmer's a priori knowledge is expressed in a first phase called the specification phase (see below, § 4.3.2). The experimental data are taken into account during an identification (or learning) phase where the free parameters of the specification are given their values.

The two robotics experiments describe above (pushing and following contours) prove the importance of this identification phase. For a given specification but different experimental data, you can obtain completely different descriptions (*i.e.* different models of the phenomenon).

4.3.2 Specification = Variables + Decomposition + Forms

The knowledge of the programmer is always expressed the same way:

1. Choose the pertinent variables.

2. Decompose the joint distribution.

3. Define the parametric forms.

This strict and simple methodology and framework for the expression of the preliminary knowledge of the programmer presents several fundamental advantages:

1. It is a programming baseline that guides any development of a Bayesian program.

2. It compels the programmer to express all available knowledge using this formalism, thus forcing a rigorous modeling process

3. It warrants that no piece of knowledge stays implicit. Everything that is known about the phenomenon is expressed within this formalism. Nothing is hidden elsewhere in any other piece of code of the program.

4. It is a formal and unambiguous common language to describe models. It could be used very efficiently to discuss and compare different models.

1. We adopt here an unambiguous *subjectivist* epistemological position about probability. Explanation about the fundamental controversy opposing *objectivism* and *subjectivism* may be found in the *FAQ/FAM 16.29 "Subjectivism vs objectivism controversy", page 203*.

5. As will be seen in the sequel, this formalism is generic and may be used to express a huge variety of models. Although simple, it offers a very strong power of expression.

4.3.3 Learning is a means to transform incompleteness into uncertainty

Descriptions are probabilistic models of a phenomenon. Descriptions are not complete. They do not escape the incompleteness curse of any nonprobabilistic model. For instance, the description of the elementary water treatment unit does not take into account the variable H which stays hidden.

However, because of learning, the influence of H is nevertheless taken into account, as its effect on the phenomenon has been captured in the values of the histograms.

Learning is a means to transform incompleteness (the effect of hidden variables) into uncertainty. The magic of this transformation is that after incompleteness has been transformed into uncertainty (probability distributions), then it is possible to reason with these distributions.

Furthermore, learning is also a means to estimate the importance of the hidden variable and consequently the quality of the model (description). If learning leads to flat distributions, it means that the neglected variables play a very important role and that the model should be improved. On the oteh hand, if learning leads to very informative (low entropy) distributions, then it confirms the quality of the model and the secondary influence of the hidden variables.

5

The Importance of Conditional Independence

What we call chance is, and may only be, the ignored cause of known effect[1].

Voltaire, Dictionaire Philosophique

The goal of this chapter is both to explain the notion of conditional independence and to demonstrate its importance in actually solving and computing complex problems.

5.1 Water treatment center Bayesian model (continuation)

In this chapter, we complete the construction of the Bayesian model for the water treatment center.

The complete process consists of four single units. Similarly, the complete Bayesian

1.Ce que nous appelons le hasard n'est, et ne peut être, que la cause ignorée d'un effet connu.

model is made of the four single models specified and identified in the previous chapter.

Putting these four models together presupposes some strong structural knowledge that can be translated into conditional independence hypotheses.

Figure 5.1 presents the functioning diagram of the water treatment center.

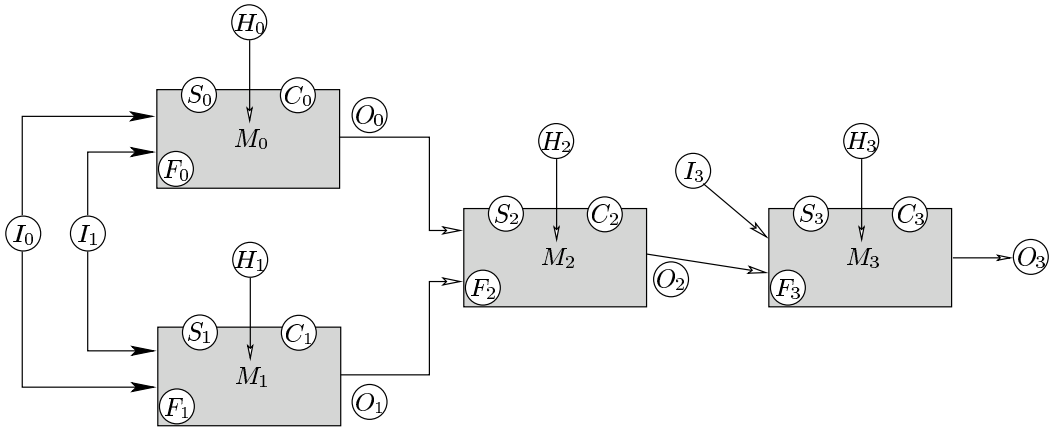

Figure 5.1: A complete water treatment center.

Units M0 and M1 take the same inputs I_0 and I_1. They respectively produce O_0 and O_1 as outputs, which in turn are used as inputs by M2. M3 takes I_3 and O_2 as inputs, and finally produces O_3 as output.

The four water treatment units have four internal states (respectively F_0, F_1, F_2, and F_3), four sensors (respectively S_0, S_1, S_2, and S_3), four controllers (respectively C_0, C_1, C_2, and C_3), and may all be perturbed by some external factor (respectively H_0, H_1, H_2, and H_3).

The production of each of these units is governed by equations (3.3) and (3.4).

The sensors take their value according to equation (3.5).

5.2 Description of the water treatment center

As in the previous chapter, to build the Bayesian model of the whole plant, we assume that H_0, H_1, H_2, and H_3 are hidden variables not known by the designer.

5.2.1 Specification

Variables

There are now 19 variables in our global Bayesian model:

$$I_0, I_1, I_3, F_0, F_1, F_2, F_3, S_0, S_1, S_2, S_3, C_0, C_1, C_2, C_3, O_0, O_1, O_2, O_3 \in \{0, ..., 10\} \quad (5.1)$$

each of these variables has a cardinality equals to 11.

Decomposition

Using the conjunction postulate (2.5) iteratively as in the previous chapter it is possible to write the joint probability on the 19 variables as:

$$\begin{aligned}
&\mathbf{P}(I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge ... \wedge O_2 \wedge O_3) \\
&= \mathbf{P}(I_0 \wedge I_1 \wedge I_3) \\
&\times \mathbf{P}(F_0 \wedge S_0 \wedge C_0 \wedge O_0 | I_0 \wedge I_1 \wedge I_3) \\
&\times \mathbf{P}(F_1 \wedge S_1 \wedge C_1 \wedge O_1 | I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0) \\
&\times \mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 | I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0 \wedge F_1 \wedge S_1 \wedge C_1 \wedge O_1) \\
&\times \mathbf{P}(F_3 \wedge S_3 \wedge C_3 \wedge O_3 | I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0 \wedge F_1 \wedge S_1 \wedge C_1 \wedge O_1 \wedge F_2 \wedge S_2 \wedge C_2 \wedge O_2)
\end{aligned} \quad (5.2)$$

This is an exact mathematical formula where we tried to regroup variables as they appeared in the four different units.

Although it is exact, this formula should obviously be further simplified! This can be done by using some additional knowledge:

1. The functioning of unit M1 depends on its entries I_0 and I_1, but is obviously independent of entry I_3 and of the operation of unit M0 (specified by variables F_0, S_0, C_0, and O_0). This leads to:

$$\begin{aligned}
&\mathbf{P}(F_1 \wedge S_1 \wedge C_1 \wedge O_1 | I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0) \\
&= \mathbf{P}(F_1 \wedge S_1 \wedge C_1 \wedge O_1 | I_0 \wedge I_1)
\end{aligned} \quad (5.3)$$

2. Operation of M2 obviously depends on the operation of M0 and M1, which produce its inputs. For instance, changing C_0 will change O_0, which in turn will

change S_2, O_2, and eventually C_2. Apparently, M2 depends on all the previous variables except I_3 and the only obvious simplification concerns I_3:

$$\begin{aligned}&\mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0 \wedge F_1 \wedge S_1 \wedge C_1 \wedge O_1)\\ &= \mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid I_0 \wedge I_1 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0 \wedge F_1 \wedge S_1 \wedge C_1 \wedge O_1)\end{aligned} \tag{5.4}$$

However, if we know the value of O_0, then we do not care anymore about the values of I_0, I_1, F_0, S_0, and C_0, which all influence the operation of M3 only by means of the output O_0. Similarly, if we know the value of O_1, then we do not care anymore about the values of F_1, S_1, and C_1. This is called *conditional independence* between variables and is a main tool to build interesting and efficient description.

One should be very careful that *conditional independence* has nothing in common with *independence*. The variable S_2 depends on C_0 ($\mathbf{P}(S_2 \mid C_0) \neq \mathbf{P}(S_2)$), but is conditionally independent of C_0 if O_0 is known ($\mathbf{P}(S_2 \mid C_0 \wedge O_0) = \mathbf{P}(S_2 \mid O_0)$). See § 5.3.1 in the sequel for further discussion of this point. This finally leads to:

$$\begin{aligned}&\mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid I_0 \wedge I_1 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0 \wedge F_1 \wedge S_1 \wedge C_1 \wedge O_1)\\ &= \mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid O_0 \wedge O_1)\end{aligned} \tag{5.5}$$

3. For the same kind of reasons, we find:

$$\begin{aligned}&\mathbf{P}(F_2 \wedge S_3 \wedge C_3 \wedge O_3 \mid I_0 \wedge I_1 \wedge I_3 \wedge F_2 \wedge F_0 \wedge S_0 \wedge C_0 \wedge O_0 \wedge F_1 \wedge S_1 \wedge C_1 \wedge O_1 \wedge F_2 \wedge S_2 \wedge C_2 \wedge O_2)\\ &= \mathbf{P}(F_3 \wedge S_3 \wedge C_3 \wedge O_3 \mid I_3 \wedge O_2)\end{aligned} \tag{5.6}$$

At this stage, we have the following decomposition:

$$\begin{aligned}
&\mathbf{P}(I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge ... \wedge O_2 \wedge O_3) \\
&= \mathbf{P}(I_0 \wedge I_1 \wedge I_3) \\
&\times \mathbf{P}(F_0 \wedge S_0 \wedge C_0 \wedge O_0 \mid I_0 \wedge I_1) \\
&\times \mathbf{P}(F_1 \wedge S_1 \wedge C_1 \wedge O_1 \mid I_0 \wedge I_1) \\
&\times \mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid O_0 \wedge O_1) \\
&\times \mathbf{P}(F_3 \wedge S_3 \wedge C_3 \wedge O_3 \mid I_3 \wedge O_2)
\end{aligned} \tag{5.7}$$

Using the same kind of assumptions as in the previous chapter, we may further simplify this expression, stating that:

4. Concerning M0:

$$\begin{aligned}
&\mathbf{P}(F_0 \wedge S_0 \wedge C_0 \wedge O_0 \mid I_0 \wedge I_1) \\
&= \mathbf{P}(F_0) \times \mathbf{P}(S_0 \mid I_0 \wedge F_0) \times \mathbf{P}(C_0) \times \mathbf{P}(O_0 \mid I_0 \wedge I_1 \wedge S_0 \wedge C_0)
\end{aligned} \tag{5.8}$$

5. Concerning M1:

$$\begin{aligned}
&\mathbf{P}(F_1 \wedge S_1 \wedge C_1 \wedge O_1 \mid I_0 \wedge I_1) \\
&= \mathbf{P}(F_1) \times \mathbf{P}(S_1 \mid I_0 \wedge F_1) \times \mathbf{P}(C_1) \times \mathbf{P}(O_1 \mid I_0 \wedge I_1 \wedge S_1 \wedge C_1)
\end{aligned} \tag{5.9}$$

6. Concerning M2:

$$\begin{aligned}
&\mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid O_0 \wedge O_1) \\
&= \mathbf{P}(F_2) \times \mathbf{P}(S_2 \mid O_0 \wedge F_2) \times \mathbf{P}(C_2) \times \mathbf{P}(O_2 \mid O_0 \wedge O_1 \wedge S_2 \wedge C_2)
\end{aligned} \tag{5.10}$$

7. Concerning M3:

$$\begin{aligned}
&\mathbf{P}(F_3 \wedge S_3 \wedge C_3 \wedge O_3 \mid I_3 \wedge O_2) \\
&= \mathbf{P}(F_3) \times \mathbf{P}(S_3 \mid I_3 \wedge F_3) \times \mathbf{P}(C_3) \times \mathbf{P}(O_3 \mid I_3 \wedge O_2 \wedge S_3 \wedge C_3)
\end{aligned} \tag{5.11}$$

After some reordering, we obtain the following final decomposition:

$$
\begin{aligned}
&\mathbf{P}(I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge \ldots \wedge O_2 \wedge O_3) \\
&= \mathbf{P}(I_0) \times \mathbf{P}(I_1) \times \mathbf{P}(I_3) \\
&\times \mathbf{P}(F_0) \times \mathbf{P}(F_1) \times \mathbf{P}(F_2) \times \mathbf{P}(F_3) \\
&\times \mathbf{P}(C_0) \times \mathbf{P}(C_1) \times \mathbf{P}(C_2) \times \mathbf{P}(C_3) \\
&\times \mathbf{P}(S_0 \mid I_0 \wedge F_0) \times \mathbf{P}(S_1 \mid I_0 \wedge F_1) \times \mathbf{P}(S_2 \mid O_0 \wedge F_2) \times \mathbf{P}(S_3 \mid I_3 \wedge F_3) \\
&\times \mathbf{P}(O_0 \mid I_0 \wedge I_1 \wedge S_0 \wedge C_0) \times \mathbf{P}(O_1 \mid I_0 \wedge I_1 \wedge S_1 \wedge C_1) \times \mathbf{P}(O_2 \mid O_0 \wedge O_1 \wedge S_2 \wedge C_2) \times \mathbf{P}(O_3 \mid I_3 \wedge O_2 \wedge S_3 \wedge C_3)
\end{aligned}
\tag{5.12}
$$

Forms

The distributions $\mathbf{P}(I_0)$, $\mathbf{P}(I_1)$, $\mathbf{P}(I_3)$, $\mathbf{P}(F_0)$, $\mathbf{P}(F_1)$,$\mathbf{P}(F_2)$,$\mathbf{P}(F_3)$, $\mathbf{P}(C_0)$, $\mathbf{P}(C_1)$, $\mathbf{P}(C_2)$, and $\mathbf{P}(C_3)$ are all assumed to be uniform distributions.

$\mathbf{P}(S_0 \mid I_0 \wedge F_0)$, $\mathbf{P}(S_1 \mid I_0 \wedge F_1)$, $\mathbf{P}(S_2 \mid O_0 \wedge F_2)$, and $\mathbf{P}(S_3 \mid I_3 \wedge F_3)$s are all Dirac distributions.

Finally, the four distributions relating the output to the inputs, the sensor, and the control are all specified as histograms, as in the previous chapter.

5.2.2 Identification

We have now four series of 11^4 histograms to identify (11^5 free parameters).

In a real control and diagnosis problem, the four production units, even though they are identical, would most probably function slightly differently (because of incompleteness and some other hidden variables besides H). In that case, the best thing to do would be to perform four different identification campaigns to take these small differences into account by learning.

In this didactic example, as the four units are simulated and formal, they really are perfectly identical (there are no possible hidden variables in our formal model as specified by equations (3.3), (3.4) and (3.5)). Consequently, we use the exact same histogram for the four units as was identified in the previous chapter.

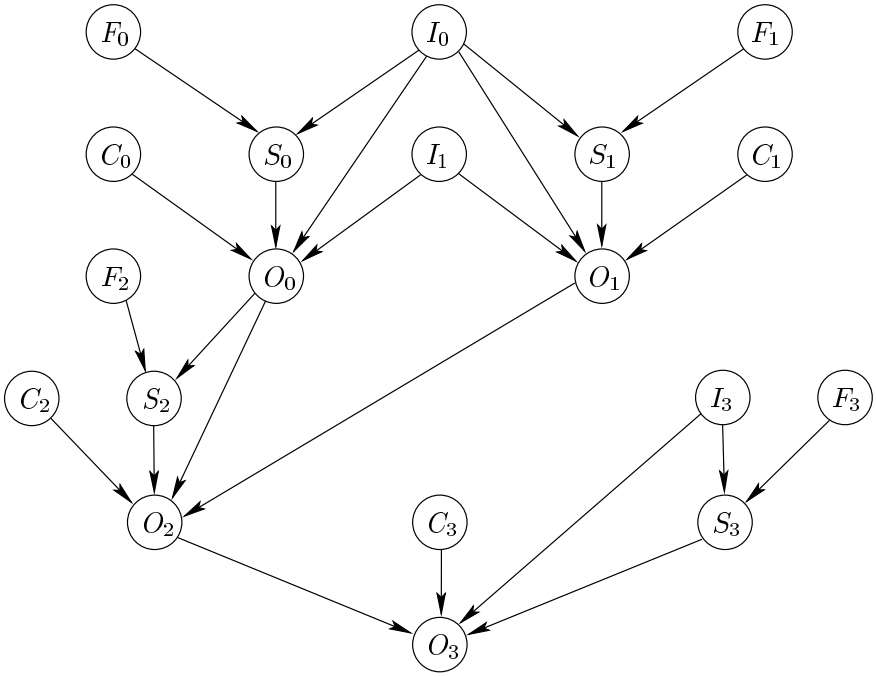

Figure 5.2: The graphical model of the joint distribution on Expression (5.12), the water treatment center shown in Figure 5.1.

5.3 Lessons, comments and notes

5.3.1 Independence versus conditional independence

To build this Bayesian model of our water treatment center we intensively used *independence* between variables and overall *conditional independence* between variables. Let us return briefly to these two concepts, which should not be confused.

Two variables X and Y are *independent* from one another if and only if $\mathbf{P}(X \wedge Y) = \mathbf{P}(X) \times \mathbf{P}(Y)$ which is logically equivalent because of the conjunction postulate to $\mathbf{P}(X \mid Y) = \mathbf{P}(X)$ and of course also to $\mathbf{P}(Y \mid X) = \mathbf{P}(Y)$.

Two variables X and Y are *independent conditionally* to a third one Z if and only if $\mathbf{P}(X \wedge Y \mid Z) = \mathbf{P}(X \mid Z) \times \mathbf{P}(Y \mid Z)$, which is also logically equivalent because of the conjunction postulate to $\mathbf{P}(X \mid Y \wedge Z) = \mathbf{P}(X \mid Z)$ and to $\mathbf{P}(Y \mid X \wedge Z) = \mathbf{P}(Y \mid Z)$.

However, two variables may be independent but not conditionally independent to a third one. Two variables may also be conditionally independent but not independent.

For the first case, in our example, for production unit M0, I_0 the first entry, and F_0 the internal state, are independent but are not conditionally independent knowing S_0 the reading of the sensor. For $I_0 = 5$ and $S_0 = 6$ Figure 5.3 shows the corresponding proba-

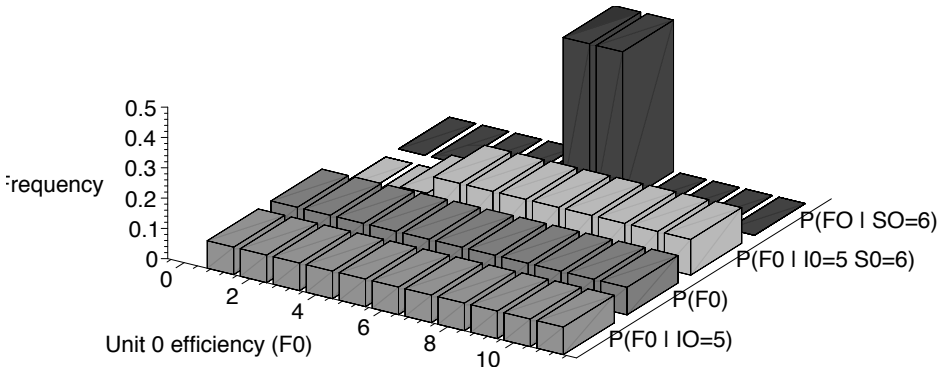

Figure 5.3: The distributions $\mathbf{P}(F_0 \mid I_0)$, $\mathbf{P}(F_0)$, $\mathbf{P}(F_0 \mid I_0 \wedge S_0)$ and $\mathbf{P}(F_0 \mid S_0)$. Note that

$$\mathbf{P}(F_0 \mid I_0) = \mathbf{P}(F_0) \text{ but } \mathbf{P}(F_0 \mid I_0 \wedge S_0) \neq \mathbf{P}(F_0 \mid S_0).$$

bilities:

This is a very common situation where the two causes of a phenomenon are independent but are conditionally dependent on one another, knowing their common consequence (see Figure 5.3). Otherwise, no sensor measuring several factors would be of any use.

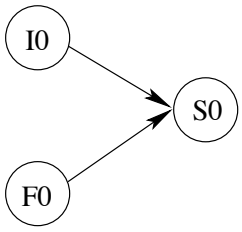

Figure 5.4: Two independent causes for the same phenomenon.

For the second case, as already said, S_2 depends on C_0 ($\mathbf{P}(S_2 \mid C_0) \neq \mathbf{P}(S_2)$) but is conditionally independent of C_0 if O_0 is known ($\mathbf{P}(S_2 \mid C_0 \wedge O_0) = \mathbf{P}(S_2 \mid O_0)$). For $C_0 = 10$ and $O_0 = 5$ Figure 5.5

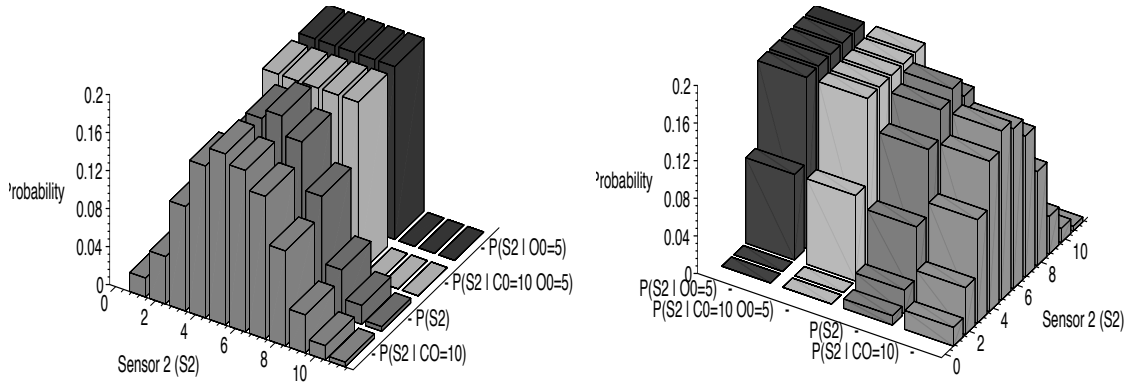

Figure 5.5: Two views of the distributions $\mathbf{P}(S_2 \mid C_0)$, $\mathbf{P}(S_2)$, $\mathbf{P}(S_2 \mid C_0 \wedge O_0)$, and $\mathbf{P}(S_2 \mid O_0)$. Note that $\mathbf{P}(S_2 \mid C_0) \neq \mathbf{P}(S_2)$ but $\mathbf{P}(S_2 \mid C_0 \wedge O_0) = \mathbf{P}(S_2 \mid O_0)$.

This is also a very common situation where there is a causal chain between three variables (see Figure 5.6).

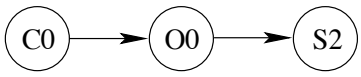

Figure 5.6: Causal chain.

5.3.2 The importance of conditional independence

This kind of conditional independence is of great importance for Bayesian Programming for two main reasons:

1. It expresses crucial designer's knowledge.

2. It makes the computation tractable by breaking the curse of dimensionality.

For any model of any phenomenon, knowing what matters with, what does not influence what and, most importantly, what bias could be neglected compared to another one is fundamental knowledge.

A model where everything depends on everything else is a very poor model, indeed. In probabilistic terms, such a model would be a joint distribution on all the relevant variables coded as a huge table containing the probabilities of all the possible cases.

In our example, simple as it is, it would be a table containing the 2^{63} probability val-

ues necessary for the joint distribution $\mathbf{P}(I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge ... \wedge O_2 \wedge O_3)$ on our 19 variables that take 11 values each:

$$11^{19} \propto 2^{63} \tag{5.13}$$

Such a table would encode all the necessary information, but in a very poor manner.

Hopefully, a model does not usually code the joint distribution in this way but rather uses a decomposition and the associated conditional independencies to express the knowledge in a structured and formal way.

The probabilistic model of the production models as expressed by equation:

$$\begin{aligned}
&\mathbf{P}(I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge ... \wedge O_2 \wedge O_3) \\
&= \mathbf{P}(I_0) \times \mathbf{P}(I_1) \times \mathbf{P}(I_3) \\
&\times \mathbf{P}(F_0) \times \mathbf{P}(F_1) \times \mathbf{P}(F_2) \times \mathbf{P}(F_3) \\
&\times \mathbf{P}(C_0) \times \mathbf{P}(C_1) \times \mathbf{P}(C_2) \times \mathbf{P}(C_3) \\
&\times \mathbf{P}(S_0 \mid I_0 \wedge F_0) \times \mathbf{P}(S_1 \mid I_0 \wedge F_1) \times \mathbf{P}(S_2 \wedge O_0 \wedge F_2) \times \mathbf{P}(S_3 \mid I_3 \wedge F_3) \\
&\times \mathbf{P}(O_0 \mid I_0 \wedge I_1 \wedge S_0 \wedge C_0) \times \mathbf{P}(0_1 \mid I_0 \wedge I_1 \wedge S_1 \wedge C_1) \times \mathbf{P}(O_2 \mid O_0 \wedge O_1 \wedge S_2 \wedge C_2) \times \mathbf{P}(O_3 \mid I_3 \wedge O_2 \wedge S_3 \wedge C_3)
\end{aligned} \tag{5.14}$$

only requires 2^{18} probability values to encode the joint distribution:

$$(11 \times 11) + (11^3 \times 4) + (11^5 \times 4) \propto 2^{18} \tag{5.15}$$

Thanks to conditional independence the curse of dimensionality has been broken!

What has been shown to be true here for the required memory space is also true for the complexity of inferences. Conditional independence is the principal tool to keep the calculation tractable. Tractability of Bayesian inference computation is of course a major concern as it has been proved NP-hard (Cooper, 1990). This subject will be developed in chapter 16 which reviews the main algorithms for Bayesian inference and is summed up in the *FAQ/FAM 16.5 "Computation complexity of Bayesian Inference", page 202*.

6

Bayesian Program = Description + Question

Far better an approximate answer to the right question which is often vague, than an exact answer to the wrong question which can always be made precise

John W. Tukey

In the two previous chapters, we built a description (Bayesian model) of our water treatment center.

In this chapter, we use this description to solve different problems: prediction of the output, choice of the best control strategy, and diagnosis of failures. This shows that multiple questions may be asked to the same description to solve very different problems. This clear separation between the model and its use is a very important feature of Bayesian Programming.

6.1 Water treatment center Bayesian model (end)

Let us first recall the present model of the process:

$$
\begin{array}{l}
\mathbf{Pr}\left\{\begin{array}{l}
\mathbf{Ds}\left\{\begin{array}{l}
\mathbf{Sp}\left\{\begin{array}{l}
\mathbf{Va} \rightarrow I_0, I_1, \ldots, O_3 \\
\mathbf{Dc} \rightarrow \left\{\begin{array}{l}
\mathbf{P}(I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge \ldots \wedge O_2 \wedge O_3) \\
= \mathbf{P}(I_0) \times \mathbf{P}(I_1) \times \mathbf{P}(I_3) \\
\times \mathbf{P}(F_0) \times \mathbf{P}(F_1) \times \mathbf{P}(F_2) \times \mathbf{P}(F_3) \\
\times \mathbf{P}(C_0) \times \mathbf{P}(C_1) \times \mathbf{P}(C_2) \times \mathbf{P}(C_3) \\
\times \mathbf{P}(S_0 \mid I_0 \wedge F_0) \times \ldots \times \mathbf{P}(S_3 \mid I_3 \wedge F_3) \\
\times \mathbf{P}(O_0 \mid I_0 \wedge I_1 \wedge S_0 \wedge C_0) \times \ldots \times \mathbf{P}(O_3 \mid I_3 \wedge 0_2 \wedge S_3 \wedge C_3)
\end{array}\right. \\
\mathbf{Fo} \rightarrow \left\{\begin{array}{l}
\mathbf{P}(I_0), \ldots, \mathbf{P}(C_3) \equiv \textbf{UNIFORM} \\
\mathbf{P}(S_0 \mid I_0 \wedge F_O), \ldots, \mathbf{P}(S_3 \mid I_3 \wedge F_3) \equiv \textbf{DIRAC} \\
\mathbf{P}(O_0 \mid I_0 \wedge I_1 \wedge S_0 \wedge C_0), \ldots, \mathbf{P}(O_3 \mid I_3 \wedge O_2 \wedge S_3 \wedge C_3) \equiv \textbf{HISTOGRAM}
\end{array}\right.
\end{array}\right. \\
\mathbf{Id}
\end{array}\right. \\
\mathbf{Qu} \rightarrow ?
\end{array}\right.
\end{array}
\tag{6.1}
$$

Quite a simple model indeed, thanks to our knowledge of this process, even if it took some time to make all its subtleties explicit!

However the question is still unspecified. Specifying and answering different possible questions is the purpose of the sequel of this chapter.

6.2 Forward simulation of a single unit

We first want to predict the output of a single unit (for instance M0) knowing its inputs, sensor reading, and control, or some of these values.

6.2.1 Question

For instance, we may look for the value of O_0 knowing the values of I_0, I_1, S_0, and C_0. The corresponding question will be:

$$\mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \ , \tag{6.2}$$

which can be computed directly as it appears in the decomposition.

This last statement can be verified by performing the corresponding computation on the joint probability distribution to check that the right simplifications occur. This computation can be broken down into different steps:

1. Application of the marginalization rule (2.10) by summing on all the missing (free) variables:

$$\begin{aligned}&\mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0)\\ &= \sum_{\substack{I_3\\ F_0 \wedge F_1 \wedge F_2 \wedge F_3\\ C_1 \wedge C_2 \wedge C_3\\ S_1 \wedge S_2 \wedge S_3\\ O_1 \wedge O_2 \wedge O_3}} \mathbf{P}(O_0 \wedge I_3 \wedge F_0 \wedge F_1 \wedge F_2 \wedge F_3 \wedge C_1 \wedge C_2 \wedge C_3 \wedge S_1 \wedge S_2 \wedge S_3 \wedge O_1 \wedge O_2 \wedge O_3 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0)\end{aligned} \tag{6.3}$$

2. Application of the conjunction postulate (2.7):

$$\begin{aligned}&\mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0)\\ &= \sum_{\substack{I_3\\ F_0 \wedge F_1 \wedge F_2 \wedge F_3\\ C_1 \wedge C_2 \wedge C_3\\ S_1 \wedge S_2 \wedge S_3\\ O1 \wedge O2 \wedge O3}} \frac{\mathbf{P}(O_0 \wedge I_3 \wedge F_0 \wedge F_1 \wedge F_2 \wedge F_3 \wedge C_1 \wedge C_2 \wedge C_3 \wedge S_1 \wedge S_2 \wedge S_3 \wedge O_1 \wedge O_2 \wedge O_3 \wedge i_0 \wedge i_1 \wedge s_0 \wedge c_0)}{\mathbf{P}(i_0 \wedge i_1 \wedge s_0 \wedge c_0)}\end{aligned} \tag{6.4}$$

3. Factorization of $\mathbf{P}(i_0 \wedge i_1 \wedge s_0 \wedge c_0)$, which is a normalization constant, as i_0, i_1, s_0, and c_0 are known values:

$$\begin{aligned}&\mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0)\\ &= \frac{1}{\Sigma} \times \sum_{\substack{I_3\\ F_0 \wedge F_1 \wedge F_2 \wedge F_3\\ C_1 \wedge C_2 \wedge C_3\\ S_1 \wedge S_2 \wedge S_3\\ O_1 \wedge O_2 \wedge O_3}} \mathbf{P}(O_0 \wedge I_3 \wedge \ldots \wedge c_0)\end{aligned} \tag{6.5}$$

4. Replacement of the joint distribution by its decomposition:

$$\mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0)$$

$$= \frac{1}{\Sigma} \times \sum_{\substack{I_3 \\ F_0 \wedge F_1 \wedge F_2 \wedge F_3 \\ C_1 \wedge C_2 \wedge C_3 \\ S_1 \wedge S_2 \wedge S_3 \\ O_1 \wedge O_2 \wedge O_3}} \begin{array}{l} \mathbf{P}(i_0 \wedge i_1 \wedge I_3) \\ \times \mathbf{P}(F_0 \wedge s_0 \wedge c_0 \wedge O_0 \mid i_0 \wedge i_1) \\ \times \mathbf{P}(F_1 \wedge S_1 \wedge C_1 \wedge O_1 \mid i_0 \wedge i_1) \\ \times \mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid O_0 \wedge O_1) \\ \times \mathbf{P}(F_3 \wedge S_3 \wedge C_3 \wedge O_3 \mid I_3 \wedge O_2) \end{array} \tag{6.6}$$

5. Reorganization and factorization of the sums:

$$\mathbf{P}(O0 \mid \mathrm{i0} \wedge \mathrm{i1} \wedge \mathrm{s0} \wedge \mathrm{c0})$$

$$= \frac{1}{\Sigma} \times \sum_{F_0} \left[\begin{array}{l} \mathbf{P}(F_0 \wedge s_0 \wedge c_0 \wedge O_0 \mid i_0 \wedge i_1) \\ \times \sum_{F_1 \wedge S_1 \wedge C_1 \wedge O_1} \left[\begin{array}{l} \mathbf{P}(F_1 \wedge S_1 \wedge C_1 \wedge O_1 \mid i_0 \wedge i_1) \\ \times \sum_{F_2 \wedge S_2 \wedge C_2 \wedge O_2} \left[\begin{array}{l} \mathbf{P}(F_2 \wedge S_2 \wedge C_2 \wedge O_2 \mid O_0 \wedge O_1) \\ \times \sum_{F_3 \wedge S_3 \wedge C_3 \wedge O_3 \wedge I_3} \left[\begin{array}{l} \mathbf{P}(F_3 \wedge S_3 \wedge C_3 \wedge O_3 \mid I_3 \wedge O_2) \\ \times \mathbf{P}(i_0 \wedge i_1 \wedge I_3) \end{array} \right] \end{array} \right] \end{array} \right] \end{array} \right] \tag{6.7}$$

6. Application of the normalization postulate (2.2), which completely simplifies the three inner sums:

$$\begin{array}{l} \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \\ = \frac{1}{\Sigma} \times \sum_{F_0} \mathbf{P}(F_0 \wedge s_0 \wedge c_0 \wedge O_0 \mid i_0 \wedge i_1) \times \mathbf{P}(i_0 \wedge i_1) \end{array} \tag{6.8}$$

7. Replacement of $\mathbf{P}(F_0 \wedge s_0 \wedge c_0 \wedge O_0 \mid i_0 \wedge i_1)$ by its decomposition (equation 5.9):

$$\begin{array}{l} \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \\ = \frac{1}{\Sigma} \times \sum_{F_0} \mathbf{P}(F_0) \times \mathbf{P}(s_0 \mid i_0 \wedge F_0) \times \mathbf{P}(c_0) \times \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \times \mathbf{P}(i_0 \wedge i_1) \end{array} \tag{6.9}$$

8. Finally, application of the normalization postulate and simplification of Σ:

$$\mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) = \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \tag{6.10}$$

The corresponding Bayesian program is:

$$\begin{array}{l}
\mathbf{Pr}\left\{\begin{array}{l}
\mathbf{Ds}\left\{\begin{array}{l}
\mathbf{Sp}\left\{\begin{array}{l}
\mathbf{Va} \rightarrow I_0, I_1, \ldots, O_3 \\
\mathbf{Dc} \rightarrow \left\{\begin{array}{l}
\mathbf{P}(I_0 \wedge I_1 \wedge I_3 \wedge F_0 \wedge \ldots \wedge O_2 \wedge O_3) \\
= \mathbf{P}(I_0) \times \mathbf{P}(I_1) \times \mathbf{P}(I_3) \\
\times \mathbf{P}(F_0) \times \mathbf{P}(F_1) \times \mathbf{P}(F_2) \times \mathbf{P}(F_3) \\
\times \mathbf{P}(C_0) \times \mathbf{P}(C_1) \times \mathbf{P}(C_2) \times \mathbf{P}(C_3) \\
\times \mathbf{P}(S_0 \mid I_0 \wedge F_0) \times \ldots \times \mathbf{P}(S_3 \mid I_3 \wedge F_3) \\
\times \mathbf{P}(O_0 \mid I_0 \wedge I_1 \wedge S_0 \wedge C_0) \times \ldots \times \mathbf{P}(O_3 \mid I_3 \wedge O_2 \wedge S_3 \wedge C_3)
\end{array}\right. \\
\mathbf{Fo} \rightarrow \left\{\begin{array}{l}
\mathbf{P}(I_0), \ldots, \mathbf{P}(C_3) \equiv \textsc{Uniform} \\
\mathbf{P}(S_0 \mid I_0 \wedge F_0), \ldots, \mathbf{P}(S_3 \mid I_3 \wedge F_3) \equiv \textsc{Dirac} \\
\mathbf{P}(O_0 \mid I_0 \wedge I_1 \wedge S_0 \wedge C_0), \ldots, \mathbf{P}(O_3 \mid I_3 \wedge O_2 \wedge S_3 \wedge C_3) \equiv \textsc{Histogram}
\end{array}\right.
\end{array}\right. \\
\mathbf{Id}
\end{array}\right. \\
\mathbf{Qu} \rightarrow \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) = \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0)
\end{array}\right.
\end{array} \tag{6.11}$$

6.2.2 Results

The corresponding results have already been presented, in Chapter 4 for $[I_0 = 2], [I_1 = 8], [C = 2]$. See Figure 4.7.

6.3 Forward simulation of the water treatment center

6.3.1 Question

We may now want to predict the output O_3 of the whole water treatment center, knowing the three inputs I_0, I_1, and I_3, the four sensor readings S_0, S_1, S_2, and S_3, and the four control values C_0, C_1, C_2, and C_3. The corresponding question is:

$$\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3) \tag{6.12}$$

This may be computed with the following computation steps:

1. Utilization of the marginalization rule (2.9):

$$\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$$

$$= \sum_{\substack{F_0 \wedge F_1 \wedge F_2 \wedge F_3 \\ O_0 \wedge O_1 \wedge O_2}} \mathbf{P}(O_3 \wedge F_0 \wedge F_1 \wedge F_2 \wedge F_3 \wedge O_0 \wedge O_1 \wedge O_2 \mid i_0 \wedge i_1 \wedge i_2 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3) \quad (6.13)$$

where we sum over all the free ($Free \equiv F_0 \wedge F_1 \wedge F_2 \wedge F_3 \wedge O_0 \wedge O_1 \wedge O_2$) variables that are neither searched ($Searched \equiv O_3$) nor known ($Known \equiv I_0 \wedge I_1 \wedge I_3 \wedge S_0 \wedge S_1 \wedge S_2 \wedge S_3 \wedge C_0 \wedge C_1 \wedge C_2 \wedge C_3$).

Equation 6.13 may then be rewritten as:

$$\begin{aligned} &\mathbf{P}(Searched \mid Known) \\ &= \sum_{Free} \mathbf{P}(Searched \wedge Free \mid Known) \end{aligned} \quad (6.14)$$

2. Utilization of the conjunction rule (2.5):

$$\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$$

$$= \sum_{F_0 \wedge \ldots \wedge O_2} \frac{\mathbf{P}(O_3 \wedge F_0 \wedge F_1 \wedge F_2 \wedge F_3 \wedge O_0 \wedge O_1 \wedge O_2 \wedge i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)}{\mathbf{P}(i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)} \quad (6.15)$$

or more succinctly:

$$\begin{aligned} &\mathbf{P}(Searched \mid Known) \\ &= \sum_{Free} \frac{\mathbf{P}(Searched \wedge Free \wedge Known)}{\mathbf{P}(Known)} \end{aligned} \quad (6.16)$$

As the entries, the sensors, and the controllers have known values, $\mathbf{P}(i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ is a constant, which by convention we name Σ. We obtain:

$$\begin{aligned} &\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3) \\ &= \frac{1}{\Sigma} \times \sum_{F_0 \wedge \ldots \wedge O_2} \mathbf{P}(O_3 \wedge F_0 \wedge \ldots \wedge O_2 \wedge i_0 \wedge \ldots \wedge c_3) \end{aligned} \quad , \quad (6.17)$$

which is equivalent to:

$$\begin{aligned}&\mathbf{P}(Searched \mid Known)\\ &= \frac{1}{\Sigma} \times \sum_{Free} \mathbf{P}(Searched \wedge Free \wedge Known)\end{aligned} \tag{6.18}$$

From the normalization rule (2.2) we have:

$$\sum_{O_3} \mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3) = 1 \tag{6.19}$$

We indeed see that, indeed, Σ is a normalization constant equal to:

$$\Sigma = \sum_{O_3} \left[\sum_{F_0 \wedge \ldots \wedge O_2} \mathbf{P} O_3 \wedge F_0 \wedge \ldots \wedge O_2 \wedge i_0 \wedge \ldots \wedge c_3 \right] \tag{6.20}$$

It is clear that whatever choice is made for *Searched* , *Known* , and *Free* we have:

$$\Sigma = \sum_{Searched} \left[\sum_{Free} \mathbf{P}(Searched \wedge Free \wedge Known) \right] \tag{6.21}$$

3. Returning to Equation (6.17) we can replace the joint probability distribution $\mathbf{P}(O_3 \wedge F_0 \wedge \ldots \wedge O_2 \wedge i_0 \wedge \ldots \wedge c_3)$ by its decomposition, give us:

$$\begin{aligned}&\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)\\ &= \frac{1}{\Sigma} \times \sum_{F_0 \wedge \ldots \wedge O_2} \left[\begin{array}{l} \mathbf{P}(i_0) \times \mathbf{P}(i_1) \times \mathbf{P}(i_2)\\ \times \mathbf{P}(F_0) \times \mathbf{P}(F_1) \times \mathbf{P}(F_2) \times \mathbf{P}(F_3)\\ \times \mathbf{P}(c_0) \times \mathbf{P}(c_1) \times \mathbf{P}(c_2) \times \mathbf{P}(c_3)\\ \times \mathbf{P}(s_0 \mid i_0 \wedge F_0) \times \ldots \times \mathbf{P}(s_3 \mid i_3 \wedge F_3)\\ \times \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \times \ldots \times \mathbf{P}(O_3 \mid i_3 \wedge O_2 \wedge s_3 \wedge c_3) \end{array} \right]\end{aligned} \tag{6.22}$$

4. Uniform distributions can be further simplified by incorporating them in the normalization constant to yield:

$$\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$$

$$= \frac{1}{\Sigma} \times \sum_{\substack{F_0 \wedge F_1 \wedge F_2 \wedge F_3 \\ O_0 \wedge O_1 \wedge O_2}} \left[\begin{array}{l} \mathbf{P}(s_0 \mid i_0 \wedge F_0) \times \ldots \times \mathbf{P}(s_3 \mid i_2 \wedge F_3) \\ \times \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \times \ldots \times \mathbf{P}(O_3 \mid i_3 \wedge O_2 \wedge s_3 \wedge c_3) \end{array} \right] \tag{6.23}$$

5. Finally, after some clever reordering of the sums to minimize the amount of computation[1], we obtain:

$$\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$$

$$= \frac{1}{\Sigma} \times \sum_{O_0} \left[\begin{array}{l} \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \\ \times \sum_{F2} \mathbf{P}(s_2 \mid O_0 \wedge F_2) \\ \times \sum_{O_1} \left[\begin{array}{l} \mathbf{P}(O_1 \mid i_0 \wedge i_1 \wedge s_1 \wedge c_1) \\ \times \sum_{O2} [\mathbf{P}(O_2 \mid O_0 \wedge O_1 \wedge s_2 \wedge c_2) \times \mathbf{P}(O_3 \mid i_3 \wedge O_2 \wedge s_3 \wedge c_3)] \end{array} \right] \end{array} \right] \tag{6.24}$$

6.3.2 Results

Some of the results obtained for the forward simulation of the water treatment center are presented in Figure 6.1.

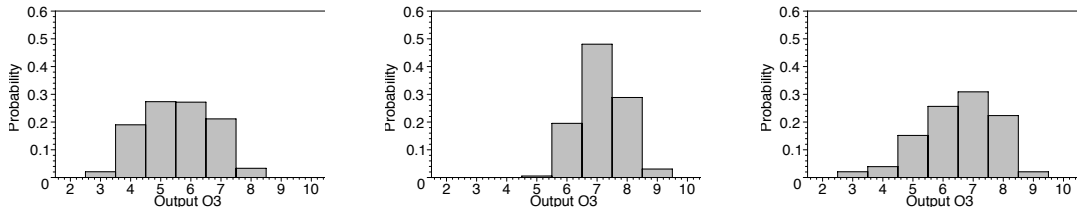

Figure 6.1: $\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ with $(i_0 = 2), (i_1 = 8), (i_2 = 10)$,$(s_0 = 5), (s_1 = 5), (s_2 = 7), (s_3 = 9)$ and three different controls: $(c_0 = c_1 = c_2 = c_3 = 0)$ (left), $(c_0 = c_1 = c_2 = c_3 = 5)$ (middle), and $(c_0 = c_1 = c_2 = c_3 = 10)$ (right).

For the same three inputs, with the same four sensor readings but with three different kinds of control we obtain completely different forecasts for the output.

This three curves show clearly that with these specific inputs and readings, an average

1. This use of reordering to minimize computation will be developed later, especially in Chapter 17

control $(c_0 = c_1 = c_2 = c_3 = 5)$, is more efficient than no control $(c_0 = c_1 = c_2 = c_3 = 0)$, or an overstated control $(c_0 = c_1 = c_2 = c_3 = 10)$.

6.4 Control of the water treatment center

Forecasting the output O_3, as in the previous section, is certainly a valuable tool to choose an appropriate control policy for the water treatment center.

However, there are $11^4 = 14{,}641$ such possible different policies. Testing all of them, one after another, would not be very practical.

Bayesian Programming endeavors to offer a direct solution to the control problem when asked the question $\mathbf{P}(C_0 \wedge C_1 \wedge C_2 \wedge C_3 \mid i_0 \wedge i_1 \wedge i_2 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge O_3)$.

6.4.1 Question (1)

After symbolic simplification and computation, the answer obtained to this question is:

$$\mathbf{P}(C_0 \wedge C_1 \wedge C_2 \wedge C_3 \mid i_0 \wedge i_1 \wedge i_2 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge O_3)$$

$$= \frac{1}{\Sigma} \times \sum_{O_0} \left[\begin{array}{l} \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge C_0) \\ \\ \times \sum_{F2} \mathbf{P}(s_2 \mid O_0 \wedge F_2) \\ \\ \times \sum_{O_1} \left[\begin{array}{l} \mathbf{P}(O_1 \mid i_0 \wedge i_1 \wedge s_1 \wedge C_1) \\ \\ \times \sum_{O_2} [\mathbf{P}(O_2 \mid O_0 \wedge O_1 \wedge s_2 \wedge C_2) \times \mathbf{P}(O_3 \mid i_3 \wedge O_2 \wedge s_3 \wedge C_3)] \end{array} \right] \end{array} \right] \quad (6.25)$$

which is basically the same expression as for the forecast of the output (see Equation (6.24) in the previous section), except that the controls are searched this time instead of being known (note the changes between the two equations between uppercase and lowercase notation for the variables).

6.4.2 Result (1)

If we fix $O_3 = 9$, the most probable control is $[C_0 = 6], [C_1 = 6], [C_2 = 4], [C_3 = 9]$.

However, the 100 best choices are very close to this best one, both in terms of probability and in terms of choice of the selected values. This means that the water treatment process is robust and that the sensivity of the solution is low.

The O_3 forecast for this control choice is presented in Figure 6.2. It shows that even

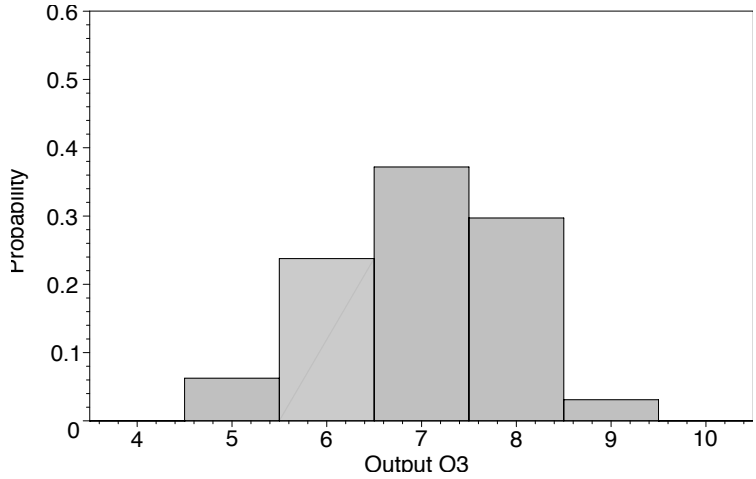

Figure 6.2: $\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ for $(i_0 = 2), (i_1 = 8), (i_3 = 10)$, $(s_0 = 5), (s_1 = 5), (s_2 = 7), (s_3 = 9)$, and $[C_0 = 6], [C_1 = 6], [C_2 = 4], [C_3 = 9]$.

if this is the best control choice, the probability of obtainig $O_3 = 9$ is only 3%. This suggest that searching the controls to find $O_3 = 9$ may still not be the best question. Indeed the present question could lead to a control choice that ensures the highest probability for $O_3 = 9$, but at the price of very high probabilities for much worse outputs.

6.4.3 Question (2)

We might rather, for instance, search for the best control strategy to ensure that $O_3 \geq o_3$. Let us, for example, analyze the cases where $o_3 \in \{4, 5, 6, 7, 8, 9\}$. That is, we need to compute

$$\mathbf{P}(C_0 \wedge C_1 \wedge C_2 \wedge C_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge O_3 \geq o_3)$$

$$= \frac{1}{\Sigma} \times \sum_{O_0} \left[\begin{array}{l} \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge C_0) \\ \\ \times \sum_{F2} \mathbf{P}(s_2 \mid O_0 \wedge F_2) \\ \\ \times \sum_{O_1} \left[\begin{array}{l} \mathbf{P}(O_1 \mid i_0 \wedge i_1 \wedge s_1 \wedge C_1) \\ \\ \times \sum_{O_2} \left[\mathbf{P}(O_2 \mid O_0 \wedge O_1 \wedge s_2 \wedge C_2) \times \sum_{O_3 \geq o_3} \mathbf{P}(O_3 \mid i_3 \wedge O_2 \wedge s_3 \wedge C_3) \right] \end{array} \right] \end{array} \right] \quad (6.26)$$

6.4.4 Result (2)

The resulting control values that maximize the probability of holding the constraint $O_3 \geq o_3$ are shown in Table 6.1. Note that, as you may have expected, the best control values for maximizing the probability of obtainnig $O_3 \geq 9$ are exactly the same as those in the previous question where we fixed $O_3 = 9$. This because the probability of $O_3 = 10$ is zero.

Constraint	Control values maximizing the probability of holding the constraint.
$O_3 \geq 4$	$\omega_4 = [C_0 = 5], [C_1 = 10], [C_2 = 10], [C_3 = 9]$
$O_3 \geq 5$	$\omega_5 = [C_0 = 5], [C_1 = 10], [C_2 = 10], [C_3 = 7]$
$O_3 \geq 6$	$\omega_6 = [C_0 = 5], [C_1 = 7], [C_2 = 6], [C_3 = 7]$
$O_3 \geq 7$	$\omega_7 = [C_0 = 5], [C_1 = 6], [C_2 = 5], [C_3 = 6]$
$O_3 \geq 8$	$\omega_8 = [C_0 = 5], [C_1 = 6], [C_2 = 4], [C_3 = 6]$
$O_3 \geq 9$	$\omega_9 = [C_0 = 6], [C_1 = 6], [C_2 = 4], [C_3 = 9]$

Table 6.1: The control values in the right column maximize the probability of holding the constraint in the left column for $\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ when $(i_0 = 2), (i_1 = 8), (i_3 = 10)$ and $(s_0 = 5), (s_1 = 5), (s_2 = 7), (s_3 = 9)$.

The distribution of $\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ for the different control values in Table 6.1 are shown below (Table 6.2) and they are depicted

graphically in Figure 6.3.

Control	$\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$										
	O_3										
	0	1	2	3	4	5	6	7	8	9	10
ω_4	0.0	0.0	0.0	0.0	0.0220845	0.0934247	0.253027	0.367386	0.242291	0.0217872	0.0
ω_5	0.0	0.0	0.0	0.0	0.0	0.038257	0.2363	0.45947	0.24463	0.021329	0.0
ω_6	0.0	0.0	0.0	0.0	0.0	0.0	0.18121	0.49079	0.29897	0.02900	0.0
ω_7	0.0	0.0	0.0	0.0	0.0	0.0	0.165783	0.503557	0.30025	0.0304099	0.0
ω_8	0.0	0.0	0.0	0.0	0.0	0.0	0.167165	0.502017	0.300179	0.0306388	0.0
ω_9	0.0	0.0	0.0	0.0	0.0	0.0624419	0.237629	0.37178	0.29711	0.0310389	0.0

Table 6.2: Probability distributions of O_3 for the different control values (ω_i) that maximize the probability of holding the constraints on Table 6.1.

Note that, the constraints $O_3 \geq 6$, $O_3 \geq 7$, and $O_3 \geq 8$ (control values ω_6, ω_7, and ω_8) give better results than the constraint $O_3 \geq 9$: there is no risk of obtaining $O_3 = 5$. The minimal output quality that we can obtain is six. This takes us to an additional question: what are the control values that maximize the *expected value* of O_3?

The expected value of a variable X is given by

$$E(X) = \sum_{x \in X} xP(x) \tag{6.27}$$

By computing the expected value of the distributions in Table 6.2 it is possible to show that the control values ω_6, ω_7, and ω_8 are a better option than ω_9. Furthermore, if we maximize the expectancy over $\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ with the fixed values $(i_0 = 2), (i_1 = 8), (i_3 = 10)$, $(s_0 = 5), (s_1 = 5), (s_2 = 7), (s_3 = 9)$ and over the space $c_0 \wedge c_1 \wedge c_2 \wedge c_3$ we obtain $\omega_* = [C_0 = 2], [C_1 = 6], [C_2 = 5], [C_3 = 6]$ with a resulting expected value of 7.20079.

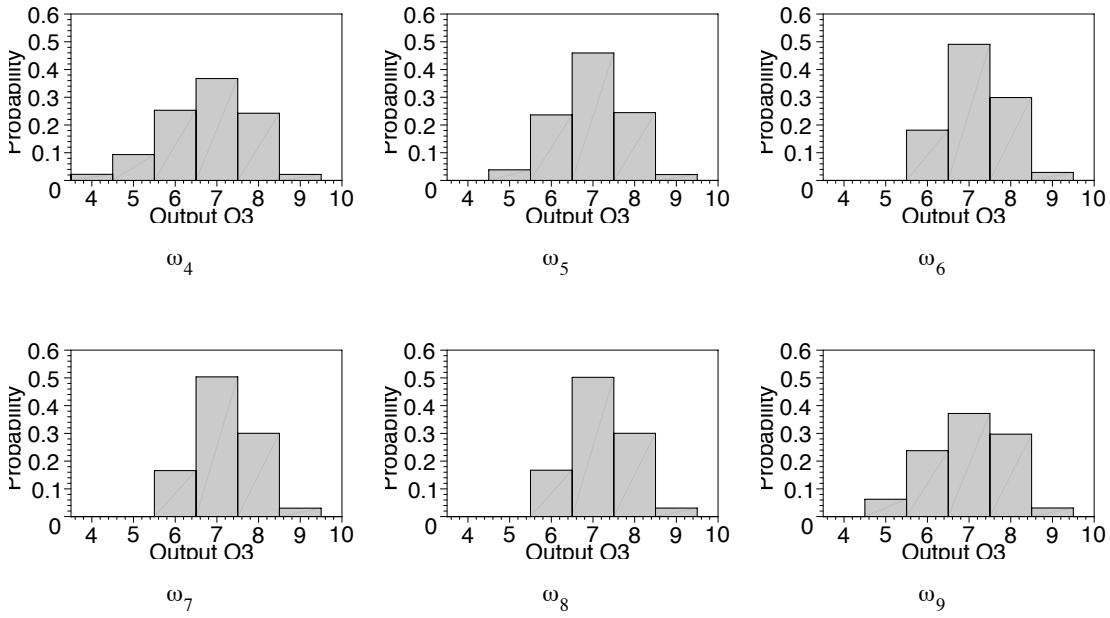

Figure 6.3: The probability distribution $\mathbf{P}(O_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ when $(i_0 = 2), (i_1 = 8), (i_3 = 10)$, $(s_0 = 5), (s_1 = 5), (s_2 = 7), (s_3 = 9)$, and variable control values $(c_0 \wedge c_1 \wedge c_2 \wedge c_3)$ are represented by ω_4 to ω_9 . These control values permit us to maximize the probability of holding the constraints in Talbe 6.1.

6.5 Diagnosis

We may also use our Bayesian model to diagnose failures.

Let us suppose that the output O_3 is only seven. This means that at least one of the four units is poor working condition. We want to identify these defective units so we can fix them.

6.5.1 Question

The question is: "what is going wrong?". We must look for the values of F_0 , F_1 , F_2 , and F_3 , knowing the entries, the sensor values, the control, and the final output.

The corresponding question is:

$$\mathbf{P}(F_0 \wedge F_1 \wedge F_2 \wedge F_3 \mid i_0 \wedge i_1 \wedge i_3 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3 \wedge o_3) \qquad (6.28)$$

which, after simplification, resolves to:

$$\mathbf{P}(F_0 \wedge F_1 \wedge F_2 \wedge F_3 \mid i_0 \wedge i_1 \wedge i_2 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3 \wedge o_3)$$

$$= \mathbf{P}(s_0 \mid i_0 \wedge F_0) \times \mathbf{P}(s_1 \mid i_0 \wedge F_1) \times \mathbf{P}(s_3 \mid i_2 \wedge F_3)$$

$$\times \sum_{O_0} \left[\begin{array}{c} \mathbf{P}(s_2 \mid O_0 \wedge F_2) \\ \times \mathbf{P}(O_0 \mid i_0 \wedge i_1 \wedge s_0 \wedge c_0) \\ \times \sum_{O_1} \left[\begin{array}{c} \mathbf{P}(O_1 \mid i_0 \wedge i_1 \wedge s_1 \wedge c_1) \\ \times \sum_{O_2} \left[\begin{array}{c} \mathbf{P}(O_2 \mid O_0 \wedge O_1 \wedge s_2 \wedge c_2) \\ \times \mathbf{P}(o_3 \mid i_2 \wedge O_2 \wedge s_3 \wedge c_3) \end{array} \right] \end{array} \right] \end{array} \right] \tag{6.29}$$

6.5.2 Results

For $O_3 = 7$, the usual entries $(i_0 = 2), (i_1 = 8), (i_3 = 10)$, sensor readings equal to $(s_0 = 5), (s_1 = 5), (s_2 = 4), (s_3 = 9)$, and controls all set to five, we require only 32 possible solutions, as presented in Table 6.3.

F_0	F_1	F_2	F_3	
8	8	2	8	0.016836
8	8	2	9	0.016836
8	8	3	8	0.051063
8	8	3	9	0.051063
8	8	4	8	0.045664
8	8	4	9	0.045664
8	8	5	8	0.011437
8	8	5	9	0.011437
8	9	2	8	0.016836
8	9	2	9	0.016836
8	9	3	8	0.051063
8	9	3	9	0.051063
8	9	4	8	0.045664
8	9	4	9	0.045664
8	9	5	8	0.011437
8	9	5	9	0.011437
9	8	2	8	0.016836
9	8	2	9	0.016836
9	8	3	8	0.051063
9	8	3	9	0.051063
9	8	4	8	0.045664
9	8	4	9	0.045664
9	8	5	8	0.011437
9	8	5	9	0.011437
9	9	2	8	0.016836
9	9	2	9	0.016836
9	9	3	8	0.051063
9	9	3	9	0.051063
9	9	4	8	0.045664
9	9	4	9	0.045664
9	9	5	8	0.011437
9	9	5	9	0.011437

Table 6.3: Results of diagnostic.

These results show clearly that the defective unit is M2, with a most probable value of three for F_2.

This can be confirmed by computing separately the probabilities for each of the four variables F_0, F_1, F_2, and F_3. F_0, F_1, and F_3 are found to have a 50% chance of being either eight or nine. In contrast, the probability distribution for F_2 is given in Figure 6.4 and gives $F_2 = 3$ as the most probable value.

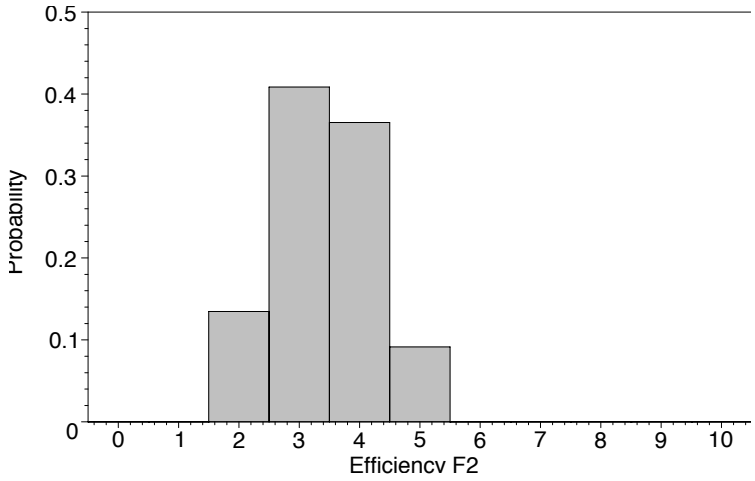

Figure 6.4: $\mathbf{P}(F_2 \mid i_0 \wedge i_1 \wedge i_2 \wedge s_0 \wedge s_1 \wedge s_2 \wedge s_3 \wedge c_0 \wedge c_1 \wedge c_2 \wedge c_3 \wedge o_3)$.

6.6 Lessons, comments and notes

6.6.1 Bayesian Program = Description + Question

A Bayesian program consists of two parts: a *description,* which is the probabilistic model of the observed phenomenon, and a *question* used to interrogate this model.

Any partition of the set of relevant variables in three subsets: the set of searched variables (which should not be empty), the set of known variables, and the complementary set of free variables, defines a valid question.

If we call *Searched* the conjunction of variables of the searched set, *Known* the conjunction of variables with known values, and *Free* the conjunction of variables in the complementary set, a question corresponds to the bundle of distributions:

$$\mathbf{P}(Searched \mid Known), \tag{6.30}$$

one distribution:

$$\mathbf{P}(Searched \mid known_i) \tag{6.31}$$

for each possible value $known_i$ of the *Known* variable.

This may be summarized by the equation: "Bayesian Program = Description + Question".

6.6.2 The essence of Bayesian inference

The essence of Bayesian inference is to be able to compute $\mathbf{P}(Searched \mid known_i)$.

This computation is always made by applying the same steps:

1. Utilization of the marginalization rule (2.9):

$$\mathbf{P}(Searched \mid known_i) = \sum_{Free} \mathbf{P}(Searched \wedge Free \mid known_i) \tag{6.32}$$

2. Utilization of the conjunction rule (2.5):

$$\mathbf{P}(Searched \mid known_i) = \sum_{Free} \frac{\mathbf{P}(Searched \wedge Free \wedge known_i)}{\mathbf{P}(known_i)} \tag{6.33}$$

3. Replacement of $\mathbf{P}(Known)$ by a normalization constant:

$$\mathbf{P}(Searched \mid known_i) = \frac{1}{\Sigma} \times \sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known_i) \tag{6.34}$$

As the *decomposition* gives us a means to compute the joint distribution $\mathbf{P}(Searched \wedge Free \wedge known_i)$ as a product of simpler distributions, we are always able to compute the wanted distribution.

However, the number of possible values for the variable *Searched* may be huge. Exhaustive and explicit computation of $\mathbf{P}(Searched \mid known_i)$ is then impossible. It must be either approximated, sampled, or used to find most probable values. In any of these three cases, the regions of interest in the searched space are the areas of high probability, which most often cover a tiny portion of the whole space. The fundamental problem is to find them, which is a very difficult optimization problem, indeed.

Worse, for each single value $searched_j$ of *Searched*, to compute $\mathbf{P}(searched_j \mid known_i)$, we need to sum on *Free* (see Equation 6.34). The number of possible values for *Free* may also be huge and the integration problem itself may be a heavy

computational burden. An approximation of the sum must be made, which means that it must effectively be computed on a sample of the whole space. Where should we sample? Obviously, to find a good estimation of the sum we should sample in areas where the probability of $\mathbf{P}(searched_j \wedge Free \wedge known_i)$ is high. This is yet another optimization problem where we must search the high-probability areas of a distribution.

All the algorithmics of Bayesian inference try to deal with these two optimization problems using various methods. A survey of these algorithms will be presented in Chapter 14. The particular solutions used in ProBT® will be described in Chapter 17.

6.6.3 No inverse or direct problem

When using a functional model defined as:

$$X = \mathbf{F}(Y) \tag{6.35}$$

computing X knowing Y is a direct problem. A search for Y knowing X is an inverse problem.

Most of the time, the direct problem is easy to solve, as we know the function $\mathbf{F}$. Often the inverse one is very difficult because $\mathbf{F}^{-1}$, the inverse function, is at best unknown and even, sometimes, nonanalitic.

Unfortunately, most of the time also, the inverse problem is much more interesting than the direct one. For instance, if $\mathbf{F}$ is the function that predicts the performance of a racing yacht, based on a knowledge of her characteristics, what is really of interest is to find the characteristics that ensures the best performance.

When using a probabilistic model defined as:

$$\mathbf{P}(X \wedge Y) \tag{6.36}$$

the difference between direct and inverse problems vanishes.

In the joint distribution, indeed, all the variables play the exact same mathematical role. Whatever the decomposition, any of the variables can in turn, in different questions, be considered as either searched, known, or free. There is no difference in the nature of the computation to calculate either $\mathbf{P}(X \mid Y)$ or $\mathbf{P}(Y \mid X)$.

However, even if any question can be asked of the probabilistic model, we stressed in the previous section that some of them can be time and resource consuming to answer.

There may be no more direct or inverse problems, but there are still some difficult problems characterized by either a huge integration space (*Free*), or a huge search space (*Searched*), or both.

6.6.4 No ill-posed problem

When using functional models, inverse problems, even when they are solvable, may be ill-posed in the sense that $Y = \mathbf{F}^{-1}(X)$ may have several solutions.

This is a very common situation in control problems, for instance, if you are given some values of the control variables, you can predict the outputs exactly, but if you are given the goal, you have numerous control solutions to achieve it.

In these cases, functional models do not have enough information to give you any hint to help you choose between the different Y satisfying $Y = \mathbf{F}^{-1}(X)$.

In contrast, $\mathbf{P}(Y \mid X)$ gives much more information, because it allows you to find the relative probabilities of the different solutions.

7

Information fusion and inverse programming

$$$

$$$

7.1 Fusion of information in ADAS systems

7.1.1 Statement of the problem

7.1.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

7.1.3 Results

7.2 Programming and training video games avatars

7.2.1 Statement of the problem

The video game industry has became during the last decade one of the most important field of the entertainment business.

As for any other kind of industry, productivity is a constant concern for the game developers. From the scriptwiter conceiving the story to the computer network engineer solving the communication and servers problems a lot of different professions are involved and huge team have to work together to produce a new game. For instance, video games make a great use of *bots*, kind of virtual robots present in the game, which populate the virtual worlds and interact with the players. These bots $$$

7.2.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

7.2.3 Results

7.3 Lessons, comments and notes

7.3.1 Information fusion

7.3.2 Coherence fusion

7.3.3 Inverse programming

The simplest way to sequence instructions or sub-programs within a program is to write them one after another. When the execution of an instruction or sub-program is complete, the computer proceeds to the next one. This mechanism supposes, first, that this sequencing is not biased by any external conditions and, second, that the end of the execution of an instruction or a sub-program is known without any uncertainty.

If the sequencing is biased by external condition, either a simple "if-then-else" or "case" mechanism is used (see Chapter 9) or, alternatively, a more sophisticated mechanism based on finite state automata. In both cases, this still supposes that both the external conditions and the end of execution are known with certainty.

Inverse programming is proposed to select sub-program sequencing when either the external conditions or the end of execution may be uncertain.

The simple ideas supporting inverse programming are: (i) to express the probability of external conditions knowing which sub-program or action is executed, (ii) to consider that these conditions are independent of one another knowing the action, and (iii) to

decide which sub-program to execute by deciding which action is the most probable, knowing the external conditions.

This sequencing mechanism has several very important qualities:

- It takes uncertainty into account;
- It requires very little memory compared to finite state automata;
- It is very efficient from a computing time point of view;
- It is very simple to specify;
- It offers easy upgrades when introducing either new conditions or new sub-programs;
- Finally, it leads to descriptions that can easily be learned from examples.

The only potential drawback of this approach is that the fundamental hypothesis that conditions are independent of one another knowing the action is far from completely satisfied in many practical cases. However, different applications of inverse programming have proved that, even though it is partially false, this hypothesis leads to satisfactory behavior. Furthermore, this hypothesis could be relaxed if necessary. This could be done by expressing in the decomposition some dependencies between the conditions, if they are really required.

8

Calling Bayesian Subroutines

$$$

The pupose of this chapter is to exhibit a first way to combine descriptions with one another in order to incrementaly build more and more complicated and sophisticated probabilistic models. This is obtained by including in the decomposition, calls to Bayesian subroutines. We show that it is hence possible to construct hierachies of probabilistic models.

8.1 Exemple 1

8.1.1 Statement of the problem

8.1.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

8.1.3 Results

8.2 Evaluation of operational risk

8.2.1 Statement of the problem

8.2.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

8.2.3 Results

8.3 Lessons, comments and notes

8.3.1 Calling subroutines

8.3.2 Hierarchies of description

9

Bayesian program mixture

$$$

$$$

9.1 Homing Behavior

9.1.1 Statement of the problem

9.1.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

9.1.3 Results

9.2 Heating forecast

9.2.1 Statement of the problem

9.2.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

9.2.3 Results

9.3 Lessons, comments and notes

9.3.1 Bayesian program combination

9.3.2 A probabilistic "if - then- else"

10

Bayesian filters

$$$

$$$

10.1 Markov localization

10.1.1 Statement of the problem

10.1.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

10.1.3 Results

10.2 ???

10.2.1 Statement of the problem

10.2.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

10.2.3 Results

10.3 Lessons, comments and notes

10.3.1 $$$

11

Using functions

$$$

$$$

11.1 ADD dice

11.1.1 Statement of the problem

11.1.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

11.1.3 Results

11.2 CAD system

11.2.1 Statement of the problem

11.2.2 Bayesian Program

Specification

Variables

Decomposition

Forms

Identification

Question

Bayesian Program

11.2.3 Results

11.3 Lessons, comments and notes

Using functions

12

Bayesian Programming Formalism

$$$

$$$

12.1 How simple! How subtle!

The purpose of this chapter is to present Bayesian Programming formally and to demonstrate that it is very simple and very clear but, nevertheless very powerful and very subtle. Probability is an extension of logic, as mathematically sane and simple as logic, but with more expressive power than logic.

It may seemed unusual to present the formalism at the end of the book. We have done this to help comprehension and to assist intuition without sacrificing rigor. After reading this chapter, anyone can check that all the examples and programs presented earlier comply with the formalism.

12.2 Logical propositions

The first concept we use is the usual notion of *logical proposition*. Propositions are denoted by lowercase names. Propositions may be composed to obtain new proposition using the usual logical operators: $a \wedge b$, denoting the conjunction of propositions a and b, $a \vee b$ their disjunction, and $\neg a$, the negation of proposition a.

12.3 Probability of a proposition

To be able to deal with uncertainty, we attach probabilities to propositions.

We consider that, to assign a probability to a proposition a, it is necessary to have at least some *preliminary knowledge*, summed up by a proposition π. Consequently, the probability of a proposition a is always conditioned, at least, by π. For each different π, $\mathbf{P}(\cdot \mid \pi)$ is an application that assigns to each proposition a a unique real value $\mathbf{P}(a \mid \pi)$ in the interval $[0, 1]$.

Of course, we are interested in reasoning about the probabilities of conjunctions, disjunctions, and negations of propositions, denoted, respectively, by $\mathbf{P}(a \wedge b \mid \pi)$, $\mathbf{P}(a \vee b \mid \pi)$ and $\mathbf{P}(\neg a \mid \pi)$.

We are also interested in the probability of proposition a conditioned by both the preliminary knowledge π and some other proposition b. This is denoted $\mathbf{P}(a \mid b \wedge \pi)$.

12.4 Normalization and conjunction postulates

Probabilistic reasoning requires only two basic rules:

1. - The *conjunction rule*, which gives the probability of a conjunction of propositions.

$$\begin{aligned} \mathbf{P}(a \wedge b \mid \pi) &= \mathbf{P}(a \mid \pi) \times \mathbf{P}(b \mid a \wedge \pi) \\ &= \mathbf{P}(b \mid \pi) \times \mathbf{P}(a \mid b \wedge \pi) \end{aligned} \tag{12.1}$$

2. - The *normalization rule*, which states that the sum of the probabilities of a and $\neg a$ is one.

$$\mathbf{P}(a \mid \pi) + \mathbf{P}(\neg a \mid \pi) = 1 \tag{12.2}$$

In this book, we take these two rules as postulates[1].

As in logic, where the resolution principle (Robinson, 1965; Robinson, 1979) is sufficient to solve any inference problem, in discrete probabilities, these two rules (12.1 & 12.2) are sufficient for any computation. Indeed, we may derive all the other necessary inference rules from these two.

12.5 Disjunction rule for propositions

For instance, the rule concerning the disjunction of propositions:

$$\begin{aligned} &\mathbf{P}(a \vee b \mid \pi) \\ &= \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) - \mathbf{P}(a \wedge b \mid \pi) \end{aligned} \tag{12.3}$$

may be derived as follows:

$$\begin{aligned} &\mathbf{P}(a \vee b \mid \pi) \\ &= (1 - \mathbf{P}(\neg a \wedge \neg b \mid \pi)) \\ &= 1 - \mathbf{P}(\neg a \mid \pi) \times \mathbf{P}(\neg b \mid \neg a \wedge \pi) \\ &= 1 - \mathbf{P}(\neg a \mid \pi) \times (1 - \mathbf{P}(b \mid \neg a \wedge \pi)) \\ &= \mathbf{P}(a \mid \pi) + \mathbf{P}(\neg a \wedge b \mid \pi) \\ &= \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) \times \mathbf{P}(\neg a \mid b \wedge \pi) \\ &= \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) \times (1 - \mathbf{P}(a \mid b \wedge \pi)) \\ &= \mathbf{P}(a \mid \pi) + \mathbf{P}(b \mid \pi) - \mathbf{P}(a \wedge b \mid \pi) \end{aligned} \tag{12.4}$$

12.6 Discrete variables

The notion of *discrete variable* is the second concept we require. Variables are denoted by names starting with one uppercase letter.

By definition, a *discrete variable* X is a set of logical propositions x_i, such that these propositions are mutually exclusive (for all i, j with $i \neq j$, $x_i \wedge x_j$ is false) and exhaustive (at least one of the propositions x_i is true). x_i means «variable X takes its i^{th} value». $\mathbf{CARD}(X)$ denotes the cardinality of the set X (the number of propositions x_i).

1.For sources giving justifications of these two rules, see *FAQ/FAM 16.6 "Cox theorem (What is?)", page 202*.

12.7 Variable conjunction

The conjunction of two variables X and Y, denoted $X \wedge Y$, is defined as the set of $\textbf{CARD}(X) \times \textbf{CARD}(Y)$ propositions $x_i \wedge y_j$. $X \wedge Y$ is a set of mutually exclusive and exhaustive logical propositions. As such, it is a new variable[1].

Of course, the conjunction of n variables is also a variable and, as such, it may be renamed at any time and considered as a unique variable in the sequel.

12.8 Probability on variables

For simplicity and clarity, we also use probabilistic formulas with variables appearing instead of propositions.

By convention, each time a variable X appears in a probabilistic formula $\Phi(X)$, it should be understood as $\forall x_i \in X, \Phi(x_i)$.

For instance, given three variables X, Y, and Z:

$$\mathbf{P}(X \wedge Y \mid Z \wedge \pi) = \mathbf{P}(X \mid Z \otimes \pi) \tag{12.5}$$

stands for:

$$\begin{gathered} \forall x_i \in X, \forall y_j \in Y, \forall z_k \in Z \\ \mathbf{P}(x_i \wedge y_j \mid z_k \wedge \pi) = \mathbf{P}(x_i \mid z_k \wedge \pi) \end{gathered} \tag{12.6}$$

12.9 Conjunction rule for variables

$$\begin{aligned} \mathbf{P}(X \wedge Y \mid \pi) &= \mathbf{P}(X \mid \pi) \times \mathbf{P}(Y \mid X \wedge \pi) \\ &= \mathbf{P}(Y \mid \pi) \times \mathbf{P}(X \mid Y \wedge \pi) \end{aligned} \tag{12.7}$$

According to our convention for probabilistic formulas including variables, this may be restated as:

1. In contrast, the disjunction of two variables, defined as the set of propositions $x_i \vee y_j$, is not a variable. These propositions are not mutually exclusive.

$$\forall x_i \in X, \forall y_j \in Y$$

$$\begin{aligned} \mathbf{P}(x_i \wedge y_j \mid \pi) &= \mathbf{P}(x_i \mid \pi) \times \mathbf{P}(y_j \mid x_i \wedge \pi) \\ &= \mathbf{P}(y_j \mid \pi) \times \mathbf{P}(x_i \mid y_j \wedge \pi) \end{aligned} \tag{12.8}$$

which may be directly deduced from the conjunction rule for propositions (12.1).

12.10 Normalization rule for variables

$$\sum_X \mathbf{P}(X \mid \pi) = 1 \tag{12.9}$$

The normalization rule may obviously be derived as follows:

$$\begin{aligned} 1 &= \mathbf{P}(x_1 \mid \pi) + \mathbf{P}(\neg x_1 \mid \pi) \\ &= \mathbf{P}(x_1 \mid \pi) + \mathbf{P}(x_2 \vee \ldots \vee x_{\mathbf{CARD}(X)} \mid \pi) \\ &= \mathbf{P}(x_1 \mid \pi) + \mathbf{P}(x_2 \mid \pi) + \ldots + \mathbf{P}(x_{\mathbf{CARD}(X)} \mid \pi) \\ &= \sum_{x_i \in X} \mathbf{P}(x_i \mid \pi) \end{aligned} \tag{12.10}$$

where the first equality derives from the normalization rule for propositions (12.2), the second from the exhaustiveness of propositions x_i, and the third from both the application of Equation (12.3) and the mutual exclusivity of propositions x_i.

12.11 Marginalization rule

$$\sum_X \mathbf{P}(X \wedge Y \mid \pi) = \mathbf{P}(Y \mid \pi) \tag{12.11}$$

The marginalization rule is derived by the successive application of the conjunction rule (12.7) and the normalization rule (12.9):

$$\begin{aligned}\sum_X \mathbf{P}(X \wedge Y \mid \pi) &= \sum_X \mathbf{P}(Y \mid \pi) \times \mathbf{P}(X \mid Y \wedge \pi) \\ &= \mathbf{P}(Y \mid \pi) \times \sum_X \mathbf{P}(X \mid Y \wedge \pi) \\ &= \mathbf{P}(Y \mid \pi)\end{aligned} \tag{12.12}$$

12.12 Bayesian program

We define a *Bayesian program* as a mean of specifying a family of probability distributions.

The constituent elements of a Bayesian program are presented in Figure 12.1:

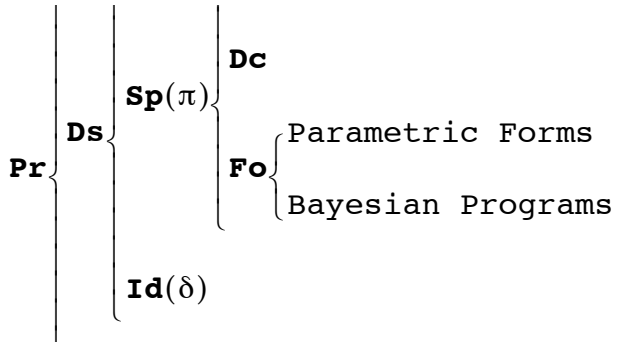

Figure 12.1: Structure of a Bayesian program

- A program is constructed from a description and a question.
- A description is constructed using some specification (π) as given by the programmer and an identification or learning process for the parameters not completely specified by the specification, using a data set (δ).
- A specification is constructed from a set of pertinent variables, a decomposition, and a set of forms.
- Forms are either parametric forms or Bayesian programs.

12.13 Description

The purpose of a description is to specify an effective method of computing a joint distribution on a set of variables $\{X_1, X_2, \ldots, X_n\}$ given a set of experimental data δ and some specification π. This joint distribution is denoted as: $\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_n \mid \delta \wedge \pi)$.

12.14 Specification

To specify preliminary knowledge, the programmer must undertake the following:

1. Define the set of relevant variables $\{X_1, X_2, \ldots, X_n\}$ on which the joint distribution is defined.

2. Decompose the joint distribution:

   Given a partition of $\{X_1, X_2, \ldots, X_n\}$ into k subsets, we define k variables $L_1, \ldots, L_k$, each corresponding to one of these subsets.

   Each variable L_i is obtained as the conjunction of the variables $\{X_{i_1}, X_{i_2}, \ldots\}$ belonging to the subset i. The conjunction rule (12.7) leads to:

$$\begin{aligned}&\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_n \mid \delta \wedge \pi) \\ &= \mathbf{P}(L_1 \mid \delta \wedge \pi) \times \mathbf{P}(L_2 \mid L_1 \wedge \delta \wedge \pi) \times \ldots \times \mathbf{P}(L_k \mid L_{k-1} \wedge \ldots \wedge L_2 \wedge L_1 \wedge \delta \wedge \pi)\end{aligned} \tag{12.13}$$

   Conditional independence hypotheses then allow further simplifications. A conditional independence hypothesis for variable L_i is defined by picking some variables X_j among the variables appearing in conjunction $L_{i-1} \wedge \ldots \wedge L_2 \wedge L_1$, calling R_i the conjunction of these chosen variables and setting:

$$\mathbf{P}(L_i \mid L_{i-1} \wedge \ldots \wedge L_2 \wedge L_1 \wedge \delta \wedge \pi) = \mathbf{P}(L_i \mid R_i \wedge \delta \wedge \pi) \tag{12.14}$$

   We then obtain:

$$\begin{aligned}&\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_n \mid \delta \wedge \pi) \\ &= \mathbf{P}(L_1 \mid \delta \wedge \pi) \times \mathbf{P}(L_2 \mid R_2 \wedge \delta \wedge \pi) \times \mathbf{P}(L_3 \mid R_3 \wedge \delta \wedge \pi) \times \ldots \times \mathbf{P}(L_k \mid R_k \wedge \delta \wedge \pi)\end{aligned} \tag{12.15}$$

   Such a simplification of the joint distribution as a product of simpler distributions is called a decomposition.

   This ensures that each variable appears at most once on the left of a conditioning bar, which is the necessary and sufficient condition to write mathematically valid decompositions.

3. Define the forms:

   Each distribution $\mathbf{P}(L_i \mid R_i \wedge \delta \wedge \pi)$ appearing in the product is then associated with either a parametric form (i.e., a function $f_\mu(L_i)$) or another Bayesian program. In general, μ is a vector of parameters that may depend on R_i or δ or both. Learning takes place when some of these parameters are computed using the data set δ.

12.15 Questions

Given a description (i.e., $\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_n \mid \delta \wedge \pi)$), a question is obtained by partitioning $\{X_1, X_2, \ldots, X_n\}$ into three sets: the searched variables, the known variables, and the free variables.

We define the variables *Searched*, *Known* and *Free* as the conjunction of the variables belonging to these sets. We define a question as the distribution:

$$\mathbf{P}(Searched \mid known \wedge \delta \wedge \pi). \tag{12.16}$$

12.16 Inference

Given the joint distribution $\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_n \mid \delta \wedge \pi)$, it is always possible to compute any possible question, using the following general inference:

$$\mathbf{P}(Searched \mid known \wedge \delta \wedge \pi) = \sum_{Free} \mathbf{P}(Searched \wedge Free \mid known \wedge \delta \wedge \pi)$$

$$= \frac{\sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known \mid \delta \wedge \pi)}{\mathbf{P}(Known \mid \delta \wedge \pi)}$$

$$= \frac{\sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known \mid \delta \wedge \pi)}{\sum_{Searched} \sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known \mid \delta \wedge \pi)} \tag{12.17}$$

$$= \frac{1}{\Sigma} \times \sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known \mid \delta \wedge \pi)$$

where the first equality results from the marginalization rule (12.11), the second results from the product rule (12.7) and the third corresponds to a second application of the marginalization rule. The denominator appears to be a normalization term. Consequently, by convention, we will replace it by Σ.

Theoretically, this allows us to solve any Bayesian inference problem. In practice, however, the cost of computing exhaustively and exactly $\mathbf{P}(Searched \mid known \wedge \delta \wedge \pi)$ is too great in most cases. Chapter 14 reviews and explains the main techniques and algorithms to deal with this inference problem.

Before that, Chapter 13 is revisiting the main Bayesian models using the present formalism.

13

Bayesian Models Revisited

$$$

The goal of this section is to present the main probabilistic models currently used for conception and development.

We systematically use the Bayesian Programming formalism to present these models, because it is precise and concise, and it simplifies their comparison.

We mainly concentrate on the definition of these models. Discussions about inference and computation are postponed to Chapter 14 and discussions about learning and identification are postponed to Chapter 15.

We chose to divide the different probabilistic models into two categories: the general purpose probabilistic models and the problem-oriented probabilistic models.

In the first category, the modeling choices are made independently of any specific knowledge about the modeled phenomenon. Most of the time, these choices are essentially made to keep the inference tractable. However, the technical simplifications of these models may be compatible with large classes of problems and consequently may have numerous applications.

In the second category, on the contrary, the modeling choices and simplifications are

decided according to some specific knowledge about the modeled phenomenon. These choices could eventually lead to very poor models from a computational wiewpoint. However, most of the time, problem-dependent knowledge such as conditional independence between variables, leads to very significant and effective simplifications and computational improvements.

13.1 General purpose probabilistic models

13.1.1 Graphical models and Bayesian networks

Bayesian networks

Bayesian networks (BNs), first introduced by Judea Pearl (Pearl, 1988), have emerged as a primary method for dealing with probabilistic and uncertain information. They are the result of the marriage between the theory of probabilities and the theory of graphs.

BNs are defined by the following Bayesian program:

$$\mathbf{Pr}\begin{cases}\mathbf{Ds}\begin{cases}\mathbf{Sp}\begin{cases}\mathbf{Va}\\ X_1, ..., X_N\\ \mathbf{Dc}\\ \mathbf{P}(X_1 \wedge ... \wedge X_N) = \prod_{i=1}^{N} \mathbf{P}(X_i \mid R_i)\\ \mathbf{Fo}\\ Any\end{cases}\\ \mathbf{Id}\end{cases}\\ \mathbf{Qu}\\ \mathbf{P}(X_i \mid Known)\end{cases} \tag{13.1}$$

- The pertinent variables are not constrained and have no specific semantics.
- The decomposition, on the contrary, is specific: it is a product of distributions with one and only one variable X_i conditioned by a conjunction of other variables R_i,

called its *parents*. An obvious bijection exists between joint probability distributions defined by such decompositions and *directed acyclic graphs* (DAG): nodes are associated with variables, and oriented edges are associated with conditional dependencies. Using graphs in probabilistic models leads to an efficient way to define hypotheses over a set of variables, an economic representation of joint probability distribution, and, most importantly, an easy and efficient way to perform probabilistic inference (see Chapter 14).

- The parametric forms are not constrained but they are very often restricted to probability tables.

- Very efficient inference techniques have been developed to answer questions $\mathbf{P}(X_i \mid known)$, however, some difficulties appear with more general questions (see Chapter 14).

Readings on Bayesian networks and graphical models should start with the following introductory textbooks: *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference (*Pearl, 1988), *Graphical Models* (Lauritzen, 1996), *Learning in Graphical Models* (Jordan, 1998) and *Graphical Models for Machine Learning and Digital Communication* (Frey, 1998).

Dynamical Bayesian networks

To deal with time and to model stochastic processes, the framework of BNs has been extended to dynamic Bayesian networks (DBNs) (see Dean & Kanazawa, 1989). Given a graph representing the structural knowledge at time t, supposing this structure to be time-invariant and time to be discrete, the resulting DBN is the repetition of the first structure from a start time to a final time. Each part at time t in the final graph is named a time slice.

They are defined by the following Bayesian program:

(13.2)

$$
\mathbf{Pr}\left\{\begin{array}{l}
\mathbf{Ds}\left\{\begin{array}{l}
\mathbf{Sp}\left\{\begin{array}{l}
\mathbf{Va}\\
X_1^0, ..., X_N^0, ..., X_1^T, ..., X_N^T\\
\mathbf{Dc}\\
\mathbf{P}(X_1^0 \wedge ... \wedge X_N^T) = \mathbf{P}(X_1^0 \wedge ... \wedge X_N^0) \times \prod_{t=0}^{T} \prod_{i=1}^{N} \mathbf{P}(X_i^t \mid R_i^t)\\
\mathbf{Fo}\\
Any
\end{array}\right.\\
\mathbf{Id}
\end{array}\right.\\
\mathbf{Qu}\\
\mathbf{P}(X_i^T \mid known)
\end{array}\right.
$$

- R_i^t is a conjunction of variables taken in the set $\left\{X_1^t, ..., X_{i-1}^t\right\} \cup \left\{X_1^{t-1}, ..., X_N^{t-1}\right\}$.

  It means that X_i^t depends only on its parents at time t ($\left\{X_1^t, ..., X_{i-1}^t\right\}$), as in a regular BN and on some variables from the previous time slice ($\left\{X_1^{t-1}, ..., X_N^{t-1}\right\}$).

- $\prod_{i=1}^{N} \mathbf{P}(X_i^t \mid R_i^t)$ defines a graph for a time slice and all time slices are identical when the time index t is changing.

- A DBN as a whole, "unrolled" over time, may be considered as a large regular BN. Consequently the usual inference techniques applicable to BNs are still valid for such "unrolled" DBNs.

The best introduction, survey and starting point on DBNs is the Ph.D. thesis of K. Murphy, *Dynamic Bayesian Networks: Representation, Inference and Learning* (Murphy, 2002).

13.1.2 Recursive Bayesian estimation: Bayesian filters, Hidden Markov Models, Kalman filters and farticle filters

Recursive Bayesian estimation: Bayesian filtering, prediction and smoothing

Recursive Bayesian estimation is the generic name for a very large applied class of probabilistic models of time series.

They are defined by the following Bayesian program:

(13.3)

$$
\begin{array}{l}
\textbf{Pr}\left\{\begin{array}{l}
\textbf{Ds}\left\{\begin{array}{l}
\textbf{Sp}\left\{\begin{array}{l}
\textbf{Va} \\
S^0, ..., S^T, O^0, ..., O^T \\
\textbf{Dc} \\
\mathbf{P}(S^0 \wedge ... \wedge S^T \wedge O^0 \wedge ... \wedge O^T) = \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{T} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)] \\
\textbf{Fo} \\
\mathbf{P}(S^0) \\
\mathbf{P}(S^i \mid S^{i-1}) \\
\mathbf{P}(O^i \mid S^i)
\end{array}\right. \\
\textbf{Id}
\end{array}\right. \\
\textbf{Qu} \\
\mathbf{P}(S^{t+k} \mid O^0 \wedge ... \wedge O^t) \qquad (k=0) \equiv Filtering \qquad (k>0) \equiv Prediction \qquad (k<0) \equiv Smoothing
\end{array}\right.
\end{array}
$$

- Variables $S^0, ..., S^T$ are a time series of state variables considered on a time horizon ranging from 0 to T. Variables $O^0, ..., O^T$ are a time series of observation variables on the same horizon.
- The decomposition is based:
  - on $\mathbf{P}(S^i \mid S^{i-1})$, called the system model, transition model or dynamic model, which formalizes the transition from the state at time i - 1 to the state at time i;

- ° on $\mathbf{P}(O^i \mid S^i)$, called the observation model, which expresses what can be observed at time i when the system is in state S^i,

- ° and on a prior $\mathbf{P}(S^0)$ over states at time 0.

- The question usually asked of these models is $\mathbf{P}(S^{t+k} \mid O^0 \wedge ... \wedge O^t)$: what is the probability distribution for the state at time t + k knowing the observations from instant 0 to t? The most common case is Bayesian filtering where $k = 0$, which means that one searches for the present state, knowing the past observations. However it is also possible to do prediction $(k > 0)$, where one tries to extrapolate a future state from past observations, or to do smoothing $(k < 0)$, where one tries to recover a past state from observations made either before or after that instant. However, some more complicated questions may also be asked (see the later section on HMM).

Bayesian filters $(k = 0)$ have a very interesting recursive property, which contributes greatly to their attractiveness. $\mathbf{P}(S^t \mid O^0 \wedge ... \wedge O^t)$ may be computed simply from $\mathbf{P}(S^{t-1} \mid O^0 \wedge ... \wedge O^{t-1})$ with the following formula:

$$\begin{aligned} &\mathbf{P}(S^t \mid O^0 \wedge ... \wedge O^t) \\ &= \mathbf{P}(O^t \mid S^t) \times \sum_{S^{t-1}} [\mathbf{P}(S^t \mid S^{t-1}) \times \mathbf{P}(S^{t-1} \mid O^0 \wedge ... \wedge O^{t-1})] \end{aligned} \tag{13.4}$$

The derivation is the following:

$$
\begin{aligned}
&= \sum_{S^0,\ldots,S^{t-1}} \left[\sum_{\substack{O^{t+1},\ldots,O^T \\ S^{t+1},\ldots,S^T}} \frac{\mathbf{P}(S^0 \wedge \ldots \wedge S^T \wedge O^0 \wedge \ldots \wedge O^T)}{\mathbf{P}(O^0 \wedge \ldots \wedge O^t)} \right] \\
&= \sum_{S^0,\ldots,S^{t-1}} \left[\sum_{\substack{O^{t+1},\ldots,O^T \\ S^{t+1},\ldots,S^T}} \frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{T} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots \wedge O^t)} \right] \\
&= \sum_{S^0,\ldots,S^{t-1}} \left[\begin{array}{l} \dfrac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots \wedge O^t)} \\ \times \sum_{\substack{O^{t+1},\ldots,O^T \\ S^{t+1},\ldots,S^T}} \left[\prod_{j=t+1}^{T} [\mathbf{P}(S^j \mid S^{j-1}) \times \mathbf{P}(O^j \mid S^j)] \right] \end{array} \right] \\
&= \sum_{S^0,\ldots,S^{t-1}} \frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots \wedge O^t)} \\
&= \mathbf{P}(O^t \mid S^t) \times \sum_{S^{t-1}} \left[\mathbf{P}(S^t \mid S^{t-1}) \times \sum_{S^0,\ldots,S^{t-2}} \frac{\mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=2}^{t-1} [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]}{\mathbf{P}(O^0 \wedge \ldots \wedge O^{t-1})} \right] \\
&= \mathbf{P}(O^t \mid S^t) \times \sum [\mathbf{P}(S^t \mid S^{t-1}) \times \mathbf{P}(S^{t-1} \mid O^0 \wedge \ldots \wedge O^{t-1})]
\end{aligned}
\tag{13.5}
$$

Another interesting aspect of this equation is to consider that there are two phases, a prediction phase and an estimation phase:

- During the prediction phase, the state is predicted using the dynamic model and the estimation of the state at the previous moment

$$
\begin{aligned}
&\mathbf{P}(S^t \mid O^0 \wedge \ldots \wedge O^{t-1}) \\
&= \sum_{S^{t-1}} [\mathbf{P}(S^t \mid S^{t-1}) \times \mathbf{P}(S^{t-1} \mid O^0 \wedge \ldots \wedge O^{t-1})]
\end{aligned}
\tag{13.6}
$$

- During the estimation phase, the prediction is either confirmed or invalidated

using the observation

$$\begin{aligned}&\mathbf{P}(S^t \mid O^0 \wedge ... \wedge O^t)\\ &= \mathbf{P}(O^t \mid S^t) \times \mathbf{P}(S^t \mid O^0 \wedge ... \wedge O^{t-1})\end{aligned} \tag{13.7}$$

Prediction $(k > 0)$ and smoothing $(k < 0)$ do not lead to such neat simplifications and tend to produce large sums that may represent a huge computational burden.

Hidden Markov Models

Hidden Markov Models (HMMs) are a very popular specialization of Bayesian filters.

They are defined by the following Bayesian program:

$$\left\{\begin{array}{l}\mathbf{Pr}\left\{\begin{array}{l}\mathbf{Ds}\left\{\begin{array}{l}\mathbf{Sp}\left\{\begin{array}{l}\mathbf{Va}\\ S^0, ..., S^t, O^0, ..., O^t\\ \mathbf{Dc}\\ \mathbf{P}(S^0 \wedge ... \wedge S^t \wedge O^0 \wedge ... \wedge O^t) = \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{t}[\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]\\ \mathbf{Fo}\\ \mathbf{P}(S^0) \equiv Matrix\\ \mathbf{P}(S^i \mid S^{i-1}) \equiv Matrix\\ \mathbf{P}(O^i \mid S^i) \equiv Matrix\end{array}\right.\\ \mathbf{Id}\end{array}\right.\\ \mathbf{Qu}\\ \mathbf{MAX}_{S^1 \wedge S^2 \wedge ... \wedge S^{t-1}}[\mathbf{P}(S^1 \wedge S^2 \wedge ... \wedge S^{t-1} \mid S^t \wedge O^0 \wedge ... \wedge O^t)]\end{array}\right.\end{array}\right. \tag{13.8}$$

- Variables are treated as discrete.

- The transition model $\mathbf{P}(S^i \mid S^{i-1})$ and the observation models $\mathbf{P}(O^i \mid S^i)$ are both specified using probability matrices.

- The question most frequently asked of HMMs is

$\mathbf{MAX}_{S^1 \wedge S^2 \wedge ... \wedge S^{t-1}}[\mathbf{P}(S^1 \wedge S^2 \wedge ... \wedge S^{t-1} \mid S^t \wedge O^0 \wedge ... \wedge O^t)]$: what is the most probable series of states that leads to the present state, knowing the past observations?[1]

This particular question may be answered with a specific and very efficient algorithm called the *Viterbi algorithm*, which is presented in Chapter 14.

A specific learning algorithm called the *Baum-Welch algorithm* has also been developed for HMMs (see Chapter 15)

A good introduction to HMMs is Rabiner's tutorial (Rabiner, 1989).

Kalman filters

The very well-known *Kalman filters* (Kalman, 1960) are another specialization of Bayesian filters.

They are defined by the following Bayesian program:

$$\mathbf{Pr}\begin{cases}\mathbf{Ds}\begin{cases}\mathbf{Sp}\begin{cases}\mathbf{Va}\\ S^0, ..., S^t, O^0, ..., O^t\\ \mathbf{Dc}\\ \mathbf{P}(S^0 \wedge ... \wedge S^t \wedge O^0 \wedge ... \wedge O^t) = \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{t}[\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(O^i \mid S^i)]\\ \mathbf{Fo}\\ \mathbf{P}(S^0) \equiv \mathbf{G}(S^0, \mu, \sigma)\\ \mathbf{P}(S^i \mid S^{i-1}) \equiv \mathbf{G}(S^i, A \bullet S^{i-1}, Q)\\ \mathbf{P}(O^i \mid S^i) \equiv \mathbf{G}(O^i, H \bullet S^i, R)\end{cases}\\ \mathbf{Id}\end{cases}\\ \mathbf{Qu}\\ \mathbf{P}(S^t \mid O^0 \wedge ... \wedge O^t)\end{cases} \tag{13.9}$$

- Variables are continuous.

1. A comon example of application of HMM models is automatic speech recognition. The states are either words or phonems and one wants to recognize the most probable sequence of states (the sentence) corresponding to the observations (the heard frequencies).

- The transition model $\mathbf{P}(S^i \mid S^{i-1})$ and the observation model $\mathbf{P}(O^i \mid S^i)$ are both specified using Gaussian laws with means that are linear functions of the conditioning variables.

With these hypotheses, and using the recursive formula (13.4), it is possible to solve the inference problem analytically to answer the usual $\mathbf{P}(S^t \mid O^0 \wedge ... \wedge O^t)$ question. This leads to an extremely efficient algorithm, which explains the popularity of Kalman filters and the number of their everyday applications.[1]

When there are no obvious linear transition and observation models, it is still often possible, using a first-order Taylor's expansion, to treat these models as locally linear. This generalization is commonly called *extended Kalman filters*.

A good tutorial by Welch and Bishop may be found on the Web (Welch & Bishop, 1997). For a more complete mathematical presentation, one should refer to a report by Barker et al. (Barker, Brown & Martin, 1994), but these are only two sources from the vast literature on this subject.

Particle filters

The fashionable particle filters may also be seen as a specific implementation of Bayesian filters.

The distribution $\mathbf{P}(S^{t-1} \mid O^0 \wedge ... \wedge O^{t-1})$ is approximated by a set of N particles having weights proportional to their probabilities. The recursive equation (13.4) is then used to inspire a dynamic process that produces an approximation of $\mathbf{P}(S^t \mid O^0 \wedge ... \wedge O^t)$. The principle of this dynamic process is that the particles are first moved according to the transition model $\mathbf{P}(S^t \mid S^{t-1})$, then their weights are updated according to the observation model $\mathbf{P}(O^t \mid S^t)$.

Arulampalam's tutorial gives a good overview of this (Arulampalam et al., 2001).

13.1.3 Mixture models

Mixture models try to approximate a distribution on a set of variables $\{X_1, ..., X_N\}$ by adding up (mixing) a set of simple distributions.

1. A very popular application of Kalman filter is the GPS (Global Positionning System). The recursive evaluation of the position explains why the precision is poor when you turn on your GPS and improves rapidely after a while. The dynamic model takes into account the previous position and speed to predict the future position (equation 13.6), when the observation model confirms (or invalidates) this position knowing the signal coming from the satelites (equation 13.7).

The most popular mixture models are Gaussian mixtures where the component distributions are Gaussian. However, the component distributions may be of any nature, for instance logistic or Poisson distributions. In the sequel, for simplicity, we will consider only Gaussian mixtures.

Such a mixture is usually defined as follows:

$$\begin{array}{l}\textbf{Pr}\\ \left|\begin{array}{l}\textbf{Ds}\\ \left\{\begin{array}{l}\textbf{Sp}\\ \left\{\begin{array}{l}\textbf{Va}\\ X_1, ..., X_N\\ \textbf{Dc}\\ \mathbf{P}(X_1 \wedge ... \wedge X_N) = \sum_{i=1}^{M} [\alpha_i \times \mathbf{P}(X_1 \wedge ... \wedge X_N \mid \pi_i)]\\ \textbf{Fo}\\ \mathbf{P}(X_1 \wedge ... \wedge X_N \mid \pi_i) \equiv \mathbf{G}(X_1 \wedge ... \wedge X_N, \mu_i, \sigma_i)\end{array}\right.\\ \textbf{Id}\end{array}\right.\\ \textbf{Qu}\end{array}\right.\end{array} \tag{13.10}$$

It should be noticed that this is not a valid Bayesian program. In particular, the decomposition does not have the right form:

$$\mathbf{P}(X_1 \wedge ... \wedge X_N) = \mathbf{P}(L^1 \mid \delta \wedge \pi) \times \prod_{i=2}^{M} \mathbf{P}(L^i \mid R^i \wedge \delta \wedge \pi) \tag{13.11}$$

It is, however, a very popular and convenient way to specify distributions $\mathbf{P}(X_1 \wedge ... \wedge X_N)$, especially when the types of the component distributions $\mathbf{P}(X_1 \wedge ... \wedge X_N \mid \pi_i)$ are chosen to ensure efficient analytical solutions to some of the inference problems.

Furthermore, it is possible to specify such a mixture as a correct Bayesian program by adding one model selection variable H to the previous definition:

$$
\mathbf{Pr}\left\{\begin{array}{l}
\mathbf{Ds}\left\{\begin{array}{l}
\mathbf{Sp}\left\{\begin{array}{l}
\mathbf{Va}\\
X_1, ..., X_N, H\\
\mathbf{Dc}\\
\mathbf{P}(X_1 \wedge ... \wedge X_N \wedge H \mid \pi)\\
= \mathbf{P}(H \mid \pi) \times \mathbf{P}(X_1 \wedge ... \wedge X_N \mid H \wedge \pi)\\
\mathbf{Fo}\\
\mathbf{P}(H \mid \pi) \equiv Table\\
\mathbf{P}(X_1 \wedge ... \wedge X_N \mid [H = i] \wedge \pi) = \mathbf{P}(X_1 \wedge ... \wedge X_N \mid \pi_i) = \mathbf{G}(X_1 \wedge ... \wedge X_N, \mu_i, \sigma_i)
\end{array}\right.\\
\mathbf{Id}\\
\mathbf{P}(H \mid \pi)
\end{array}\right.\\
\mathbf{Qu}\\
\mathbf{P}(Searched \mid known)\\
= \sum_H \mathbf{P}(X_1 \wedge ... \wedge X_N \wedge H) = \sum_{i=1}^{M} [\mathbf{P}([H = i] \mid \pi) \times \mathbf{P}(X_1 \wedge ... \wedge X_N \mid \pi_i)]
\end{array}\right. \tag{13.12}
$$

- H is a discrete variable, taking M values. H is used as a selection variable. Knowing the value of H, we suppose that the joint distribution is reduced to one of its component distributions:

$$\mathbf{P}(X_1 \wedge ... \wedge X_N \mid [H = i] \wedge \pi) = \mathbf{P}(X_1 \wedge ... \wedge X_N \mid \pi_i) \tag{13.13}$$

- In these simple and common mixture models, H, the mixing variable is assumed to be independent of the other variables. We saw in Chapter 9 a discussion of more elaborate mixing models where H depends on some of the other variables. This is also the case in expert mixture models, as described by Jordan (see Jordan & Jacobs, 1994 and Meila & Jordan, 1996)

- Identification is a crucial step for these models, when the values of $\mathbf{P}(H \mid \pi)$ and the parameters $\mu_1, ..., \mu_M, \sigma_1, ..., \sigma_M$ of the component distributions are searched to find the best possible fit between the observed data and the joint distribution. This is usually done using the EM algorithm or some of its variants (see Chapter 15).

- The questions asked of the joint distribution are of the form $\mathbf{P}(Searched \mid known)$

where *Searched* and *Known* are conjunctions of some of the $X_1, ..., X_N$. The parameters $\mu_1 \wedge ... \wedge \mu_M \wedge \sigma_1 \wedge ... \wedge \sigma_M$ of the component distributions are known, but the selection variable H is unknown, as it always stays hidden. Consequently, solving the question assumes a summation for the possible value of H, and we finally retrieve the usual mixture form:

$$\begin{aligned} &\mathbf{P}(Searched \mid known) \\ &= \sum_{i=1}^{M} [\mathbf{P}([H = i] \mid \pi) \times \mathbf{P}(X_1 \wedge ... \wedge X_N \mid \pi_i)] \end{aligned} \tag{13.14}$$

A reference on mixture models is McLachlan's book *Finite Mixture Models (*McLachlan & Deep, 2000*)*.

13.1.4 Maximum entropy approaches

Maximum entropy approaches play a very important role in physical applications. The late E.T. Jaynes, in his regrettably unfinished book (Jaynes, 2003), gives a wonderful presentation of them as well as a fascinating apologia for the subjectivist epistemology of probabilities.

The maximum entropy models may be described by the following Bayesian program:

$$\mathbf{Pr}\left\{\begin{array}{l}\mathbf{Ds}\left\{\begin{array}{l}\mathbf{Sp}\left\{\begin{array}{l}\mathbf{Va}\\ X_1, ..., X_N\\ \mathbf{Dc}\\ \mathbf{P}(X_1 \wedge ... \wedge X_N)\\ = \prod_{i=0}^{M}[e^{-[\lambda_i \times \mathbf{f_i}(X_1 \wedge ... \wedge X_N)]}] = e^{-\sum_{i=0}^{M}[\lambda_i \times \mathbf{f_i}(X_1 \wedge ... \wedge X_N)]}\\ \mathbf{Fo}\\ \mathbf{f}_0 = 1 \qquad \mathbf{f}_1, ..., \mathbf{f}_M \text{ M observable functions}\end{array}\right.\\ \mathbf{Id}\\ \{\lambda_1, ..., \lambda_M\}\end{array}\right.\\ \mathbf{Qu}\\ \mathbf{P}(Searched \mid Known)\end{array}\right. \tag{13.15}$$

- The variables $X_1, ..., X_N$ are not constrained.

- The decomposition is made of a product of exponential distributions $e^{-[\lambda_i \times \mathbf{f_i}(X_1 \wedge ... \wedge X_N)]}$ where each $\mathbf{f_i}$ is called an *observable function*. An observable function may be any real function on the space defined by $X_1 \wedge ... \wedge X_N$, such that its expectation may be computed:

$$\langle \mathbf{f_i}(X_1 \wedge ... \wedge X_N) \rangle = \sum_{X_1 \wedge ... \wedge X_N} [\mathbf{P}(X_1 \wedge ... \wedge X_N) \times \mathbf{f_i}(X_1 \wedge ... \wedge X_N)] \tag{13.16}$$

- The constraints on the problem are usually expressed by M real values F_i called *levels of constraint*, which impose the condition $\langle \mathbf{f_i}(X_1 \wedge ... \wedge X_N) \rangle = F_i$.

  These levels of constraint may either be arbitrary values fixed *a priori* by the programmer, or the results of experimental observation. In this latter case, the levels of constraint are equal to the observed mean values of the observable functions on the data set. The constraint then imposed on the distribution is that the expectations of the observable functions according to the distribution should be equal to the means of these observable functions.

- The identification problem is, then, knowing the level of constraint F_i, to find the *Lagrange multipliers* λ_i that maximize the entropy of the distribution $\mathbf{P}(X_1 \wedge ... \wedge X_N)$.

The maximum entropy approach is a very general and powerful way to represent probabilistic models and to explain what is going on when one wants to identify the parameters of a distribution, choose its form, or even compare models. Unfortunately, finding the values of the Lagrange multipliers λ_i can be very difficult.

A sound introduction is of course Jaynes' book *Probability Theory - The Logic of Science* (Jaynes, 2003). Other references are the edited proceedings of the regular MaxEnt conferences, which cover both theory and applications (see Levine & Tribus, 1979; Erickson & Smith, 1988a; Erickson & Smith, 1988b; Kapur & Kesavan, 1992; Smith & Grandy, 1985; Mohammad-Djafari & Demoment, 1992 and Mohammad-Djafari, 2000).

13.2 Problem-oriented probabilistic models

13.2.1 Sensor fusion

Sensor fusion is a very common and crucial problem for both living systems and artifacts. The problem is as follows: given a phenomenon and some sensors, how can we derive information on the phenomenon by combining the information from the different sensors?

The most common and simple Bayesian modeling for sensor fusion is the following:

$$\mathbf{Pr}\left\{\begin{array}{l}\mathbf{Ds}\left\{\begin{array}{l}\mathbf{Sp}\left\{\begin{array}{l}\mathbf{Va}\\ \Phi, S_1, ..., S_N\\ \mathbf{Dc}\\ \mathbf{P}(\Phi \wedge S_1 \wedge ... \wedge S_N) = \mathbf{P}(\Phi) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi)\\ \mathbf{Fo}\\ Any\end{array}\right.\\ \mathbf{Id}\end{array}\right.\\ \mathbf{Qu}\\ \mathbf{P}(\Phi \mid S_1 \wedge ... \wedge S_N) = \frac{1}{Z} \times \mathbf{P}(\Phi) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi)\\ \mathbf{P}(Searched \mid Known)\end{array}\right. \tag{13.17}$$

- Φ is the variable used to describe the phenomenon, when $\{S_1, ..., S_N\}$ are the variables encoding the readings of the sensors.

- The decomposition:

$$\mathbf{P}(\Phi \wedge S_1 \wedge ... \wedge S_N) = \mathbf{P}(\Phi) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi) \tag{13.18}$$

  may seem peculiar, as the readings of the different sensors are obviously not independent from one another. The exact meaning of this equation is that the phenomenon Φ is considered to be the main reason for the contingency of the readings. Consequently, it is stated that knowing Φ, the readings S_i are independent. Φ is the cause of the readings and, knowing the cause, the consequences are independent. Indeed, this is a very strong hypothesis, far from always being satisfied. However, it very often gives satisfactory results and has the main advantage of considerably reducing the complexity of the computation.

- The distributions $\mathbf{P}(S_i \mid \Phi)$ are called *sensor models*. Indeed, these distributions encode the way a given sensor responds to the observed phenomenon. When dealing with industrial sensors, this is the kind of information directly provided by the

device manufacturer. However, these distributions may also be identified very easily by experiment.

- The most common question asked of this fusion model is $\mathbf{P}(\Phi \mid S_1 \wedge ... \wedge S_N)$. It should be noticed that this is an inverse question. The capacity to answer such inverse questions easily is one of the main advantages of probabilistic mode ling. This is not the only possible question: any question may be asked of this kind of model, and the decomposition generally leads to tractable computations. For instance:

  ° $\mathbf{P}(\Phi \mid S_i \wedge S_j) = \frac{1}{Z} \times \mathbf{P}(\Phi) \times \mathbf{P}(S_i \mid \Phi) \times \mathbf{P}(S_j \mid \Phi)$ makes the fusion of two single sensors. It simplifies neatly as the product of the two corresponding sensor models.

  ° $\mathbf{P}(S_k \mid \Phi \wedge S_i \wedge S_j) = \frac{1}{Z} \times \mathbf{P}(\Phi) \times \mathbf{P}(S_i \mid \Phi) \times \mathbf{P}(S_j \mid \Phi) \times \mathbf{P}(S_k \mid \Phi)$ tries to measure the coherence between the readings of three sensors and may be used to diagnose the failure of the k^{th} sensor.

13.2.2 Classification

The classification problem may be seen as the same as the sensor fusion problem just described. Usually, the problem is called a classification problem when the possible value for Φ is limited to a small number of classes and it is called a sensor fusion problem when Φ can be interpreted as a "measure".

A slightly more subtle definition of classification uses one more variable. In this model, not only is there the variable Φ, used to merge the information, but there is C, used to classify the situation. C has far less values than Φ and it is possible to specify $\mathbf{P}(\Phi \mid C)$, which, for each class makes the possible values of Φ explicit. Answering the classification question $\mathbf{P}(C \mid S_1 \wedge ... \wedge S_N)$ supposes a summation over the different values of Φ.

The Bayesian program then obtained is as follows:

(13.19)

$$
\begin{array}{l}
\mathbf{Pr}\left\{\begin{array}{l}
\mathbf{Ds}\left\{\begin{array}{l}
\mathbf{Sp}\left\{\begin{array}{l}
\mathbf{Va} \\
C, \Phi, S_1, ..., S_N \\
\mathbf{Dc} \\
\mathbf{P}(C \wedge \Phi \wedge S_1 \wedge ... \wedge S_N) = \mathbf{P}(C) \times \mathbf{P}(\Phi \mid C) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi) \\
\mathbf{Fo} \\
Any
\end{array}\right. \\
\mathbf{Id}
\end{array}\right. \\
\mathbf{Qu} \\
\mathbf{P}(C \mid S_1 \wedge ... \wedge S_N) = \frac{1}{Z} \times \sum_{\Phi} \left[\mathbf{P}(C) \times \mathbf{P}(\Phi \mid C) \times \prod_{i=1}^{N} \mathbf{P}(S_i \mid \Phi) \right]
\end{array}\right.
\end{array}
$$

13.2.3 Pattern recognition

Pattern recognition is another form of the same problem as the two preceding ones. However, it is called *recognition* because the emphasis is put on deciding a given value for C rather than finding the distribution $\mathbf{P}(C \mid S_1 \wedge ... \wedge S_N)$.

Consequently, the pattern recognition community usually does not make a clear separation between the probabilistic inference part of the reasoning and the decision part, using a utility function. Both are considered as a single and integrated decision process.

The best reference work on pattern recognition is still Duda's book *Pattern Classification and Scene Analysis* (Duda & Hart, 1973).

13.2.4 Sequence recognition

The problem is to recognize a sequence of states knowing a sequence of observations and, possibly, a final state.

In Section 13.1.2 we presented Hidden Markov Models (HMMs) as a special case of Bayesian filters. These HMMs have been specially designed for sequence recognition, which is why the most common question asked of these models is $\mathbf{P}(S^1 \wedge S^2 \wedge ... \wedge S^{t-1} \mid S^t \wedge O^0 \wedge ... \wedge O^t)$ (see Equation 13.8). That is also why the Vit-

erbi algorithm, a specialized inference algorithm, has been conceived to answer this specific question (see Chapter 14).

13.2.5 Markov localization

Another possible variation of the Bayesian filter formalism is to add a control variable A to the system. This extension is sometimes called an input-output HMM (Bengio & Frasconi, 1995, Cacciatore & Nowlan, 1994, Ghahramani, 2001, Meila & Jordan, 1996). However, in the field of robotics, it has received more attention under the name of Markov localization (Burgard et al., 1996, Thrun, Burgard & Fox, 1998). In this field, such an extension is natural, as the robot can observe its state by sensors, but can also influence its state via motor commands.

Starting from a Bayesian filter structure, the control variable is used to refine the transition model $\mathbf{P}(S^i \mid S^{i-1})$ of the Bayesian filter into $\mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1})$, which is then called the *action model*. The rest of the Bayesian filter is unchanged. The Bayesian program then obtained is as follows:

$$
\mathbf{Pr}\left\{\begin{array}{l}
\mathbf{Ds}\left\{\begin{array}{l}
\mathbf{Sp}\left\{\begin{array}{l}
\mathbf{Va} \\
S^0, ..., S^t, A^0, ..., A^{t-1}, O^0, ..., O^t \\
\mathbf{Dc} \\
\mathbf{P}(S^0 \wedge ... \wedge S^t \wedge A^0 \wedge ... \wedge A^{t-1}) \\
= \mathbf{P}(S^0) \times \mathbf{P}(O^0 \mid S^0) \times \prod_{i=1}^{t} [\mathbf{P}(A^{i-1}) \times \mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1}) \times \mathbf{P}(O^i \mid S^i)] \\
\mathbf{Fo} \\
Tables
\end{array}\right. \\
\mathbf{Id}
\end{array}\right. \\
\mathbf{Qu} \\
\mathbf{P}(S^t \mid A^0 \wedge ... \wedge A^{t-1} \wedge O^0 \wedge ... \wedge O^t) \\
= \frac{1}{Z} \times \mathbf{P}(A^{t-1}) \times \mathbf{P}(O^t \mid S^t) \times \sum_{S^{t-1}} [\mathbf{P}(S^t \mid S^{t-1} \wedge A^{t-1}) \times \mathbf{P}(S^{t-1} \mid A^0 \wedge ... \wedge A^{t-2} \wedge O^0 \wedge ... \wedge O^{t-1})]
\end{array}\right. \tag{13.20}
$$

The resulting model is used to answer the question $\mathbf{P}(S^t \mid A^0 \wedge ... \wedge A^{t-1} \wedge O^0 \wedge ... \wedge O^t)$, which estimates the state of the robot, given past actions and observations. When this state represents the position of the robot in its envi-

ronment, this amounts to localization.

A reference for Markov localization and its use in robotics is Thrun's survey *Probabilistic Algorithms in Robotics* (Thrun, 2000).

13.2.6 Markov decision processes

Partially observable Markov decision processes (POMDPs)

POMDPs are used to model a robot that must plan and execute a sequence of actions.

Formally, POMDPs use the same probabilistic model as Markov localization except that they are enriched by the definition of a *reward* (and/or cost) function.

This reward function $\mathbf{R}$ models those states that are good for the robot, and which actions are costly. In the most general notation, it is therefore a function that associates, for each state-action couple a real-valued number: $\mathbf{R}, S^i \wedge A^i \rightarrow \Re$.

The reward function also helps to drivr the planning process. Indeed, the aim of this process is to find an optimal plan in the sense that it maximizes a certain measure based on the reward function. This measure is most frequently the expected discounted cumulative reward:

$$\left\langle \sum_{t=0}^{\infty} \gamma^t \times R^t \right\rangle \tag{13.21}$$

where γ is a discount factor (less than 1), R^t is the reward obtained at time t, and $\langle\ \rangle$ is the mathematical expectation. Given this measure, the goal of the planning process is to find an optimal mapping from probability distributions over states to actions (a *policy*). This planning process, which leads to intractable computation, is sometimes approximated using iterative algorithms called policy iteration or value iteration. These algorithms start with random policies, and improve them at each step until some numerical convergence criterion is met. Unfortunately, state-of-the-art implementations of these algorithms still cannot cope with state spaces of more than 100 states (Pineau & Thrun, 2002).

An introduction to POMDPs is provided by Kaelbling et al. (Kaelbling, Littman & Cassandra, 1998).

Markov decision process

Another approach for tackling the intractability of the planning problem in POMDPs is to suppose that the robot knows what state it is in. The state becomes observable, therefore the observation variable and model are no longer needed; the resulting formalism is called a (fully observable) MDP, and is summarized by the following Bayesian program:

(13.22)

$$
\mathbf{Pr}\left\{\begin{array}{l}
\mathbf{Ds}\left\{\begin{array}{l}
\mathbf{Sp}\left\{\begin{array}{l}
\mathbf{Va} \\
S^0, ..., S^t, A^0, ..., A^{t-1} \\
\mathbf{Dc} \\
\mathbf{P}(S^0 \wedge ... \wedge S^t \wedge A^0 \wedge ... \wedge A^{t-1}) = \mathbf{P}(S^0) \times \prod_{i=1}^{t} [\mathbf{P}(A^{i-1}) \times \mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1})] \\
\mathbf{Fo} \\
Tables
\end{array}\right. \\
\mathbf{Id}
\end{array}\right. \\
\mathbf{Qu} \\
\mathbf{P}(A^0 \wedge ... \wedge A^{t-1} \mid S^t \wedge S^0)
\end{array}\right.
$$

- The variables are: $\{S^0, ..., S^t\}$ a temporal sequence of states and $\{A^0, ..., A^{t-1}\}$ a temporal sequence of actions.

- The decomposition makes a first-order Markov assumption by specifying that the state at time t depends on the state at time t - 1 and also on the action taken at time t - 1.

- $\mathbf{P}(S^i \mid S^{i-1} \wedge A^{i-1})$ is usually represented by a matrix and is called the "transition matrix" of the model.

- The question addressed to the MDP is:

$$\mathbf{P}(A^0 \wedge ... \wedge A^{t-1} \mid S^t \wedge S^0) \tag{13.23}$$

What is the sequence of actions required to go from state S^0 to state S^t?

MDPs can cope with planning in state-spaces bigger than POMDPs, but are still limited to some hundreds of states. Therefore, recently many research efforts have been aimed toward hierarchical decomposition of the planning and modeling problems in MDPs, especially in robotics, where the full observability hypothesis makes their practical use difficult (Hauskrecht et al., 1998; Lane & Kaelbling, 2001; Pineau & Thrun, 2002 & Diard, 2003).

A complete review of POMDPs and MDPs by Boutilier et al. (Boutilier, Dean & Hanks, 1999) is an interesting complement.

13.2.7 Bayesian models in life science

Finally, in life science and especially in neuroscience and psychology, the application of Bayesian models is quickly becoming widespread.

The main questions of interest are the following:

- At the microscopic level, is there any evidence that a neuron or an assembly of neurons may be performing Bayesian inference and learning?

- At a macroscopic level, can we explain the results of psychophysical experiments with Bayesian models, respecting the constraints of physiology?

There is significant literature on these subjects. The paper entitled *The exploitation of regularities in the environment by the brain* (Barlow, 2001) gives an interesting historical perspective of these questions. The following books offer an initial overview of this literature: *Perception as Bayesian Inference* (Knill & Richards, 1996), *The Mind's Arrows: Bayes Nets and Graphical Causal Models in Psychology* (Glymour, 2001), *Graphical Models: Foundations of Neural Computation* (Jordan & Sejnowski, 2001), *Theoretical Neuroscience* (Dayan & Abbott, 2001), *Probabilistic Models of the Brain* (Poincaré, H (1902) La science et l'hypothèse; Originally published in 1902, éditions de la Bohème, Rueil-Malmaison, FranceRao, Olshausen & Lewicki, 2002), and *Decisions, Uncertainty, and the Brain: The Science of Neuroeconomics* (Glimcher, 2003).

13.3 Summary

The main hierarchical relationships between these primary models are outlined in Figure 13.1 below. The more general models appear on the left, whith the more specific on the right. An arrow connects a model to one of its specializations.

Figure 13.1: Hierarchical interdependencies between principal Bayesian models.

14

Bayesian Inference Algorithms Revisited

$$$

$$$

14.1 Stating the problem

Given the joint distribution:

$$\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_n) = \mathbf{P}(L_1) \times \mathbf{P}(L_2 \mid R_2) \times \mathbf{P}(L_3 \mid R_3) \times \ldots \times \mathbf{P}(L_k \mid R_k) \quad (14.1)$$

it is always possible to compute any possible question $\mathbf{P}(\mathit{Searched} \mid \mathit{known})$ using the following general inference:

$$
\begin{aligned}
&\overline{Free} \\
&= \frac{\sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known)}{\mathbf{P}(known)} \\
&= \frac{\sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known)}{\sum_{Searched} \left[\sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known) \right]} \\
&= \frac{1}{\Sigma} \times \sum_{Free} \mathbf{P}(Searched \wedge Free \wedge known) \\
&= \frac{1}{\Sigma} \times \sum_{Free} \left[\mathbf{P}(L_1) \times \prod_{i}^{k} \mathbf{P}(L_i \mid R_i) \right]
\end{aligned}
\tag{14.2}
$$

where the first equality results from the marginalization rule (12.11), the second results from the conjunction rule (12.7), and the third corresponds to a second application of the marginalization rule. The denominator appears to be a normalization term. Consequently, by convention, we will replace it with Σ. Finally, it is possible to replace the joint distribution by its decomposition (12.15).

It is well known that general Bayesian inference is a very difficult problem, which may be practically intractable. Exact inference has been proved to be NP-hard (Cooper, 1990), as has the general problem of approximate inference (Dagum & Luby, 1993).

Numerous heuristics and restrictions to the generality of the possible inferences have been proposed to achieve admissible computation time. The purpose of this chapter is to provide a short review of these heuristics and techniques.

Before starting to crunch any numbers, it is usually possible (and wise) to make some symbolic computations to reduce the amount of numerical computation required.

The problem can be stated very simply. How can we modify the expression:

$$
\frac{1}{\Sigma} \times \sum_{Free} \left[\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i) \right] \tag{14.3}
$$

to produce a new expression requiring less computation that gives the same result or a

good approximation to it?

Section 14.2 presents the different possibilities. We will see that these symbolic computations can be either exact (Section 14.2.1) or approximate (Section 14.2.2), in which case they lead to an expression that, while not mathematically equal to Equation (14.3), should be close enough.

Once simplified, the expression obtained to compute $\mathbf{P}(Searched \mid known)$ must be evaluated numerically.

In a few cases, exact (exhaustive) computation may be possible, thanks to the previous symbolic simplification, but normally, even with the simplifications, only approximate calculation is possible. Section 14.3 describes the main algorithms used.

Two main problems must be solved: searching the modes in a high-dimensional space, and marginalizing in a high-dimensional space.

Because *Searched* may be a conjunction of numerous variables, each of them possibly having many values or even being continuous, it is seldom possible to compute exhaustively $\mathbf{P}(Searched \mid known)$ and find the absolute most probable value for *Searched*. One may then decide either to build an approximate representation of this distribution or to directly sample from this distribution. In both cases, the challenge is to find the modes of:

$$\mathbf{P}(Searched \mid known) \propto \sum_{Free} \left[\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i) \right] \tag{14.4}$$

(on the search space defined by *Searched*), where most of the probability density is concentrated. This may be very difficult, as most of the probability may be concentrated in very small subspaces of the whole search space. Searching the modes of a distribution in a high-dimensional space will be the subject of Section 14.3.1.

The situation is even worse, as computing the value of $\mathbf{P}(Searched \mid known)$ for a given value of *Searched* (a single point of the search space of the preceding paragraph) is by itself a difficult problem. Indeed, it requires marginalizing the joint distribution on the space defined by *Free*. *Free* (like *Searched*) may be a conjunction of numerous variables, each of them possibly having many values or even being continuous. Consequently, the sum should also be either approximated or sampled. The challenge is then to find the modes of

$$\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i) \tag{14.5}$$

(on the search space defined by *Free*), where most of the probability density is concentrated and which mostly contribute to the sum. Finally, marginalizing in a high-dimensional space appears to be a very similar problem to searching the modes in a high-dimensional space. This appears clearly in Section 14.3.2, which briefly deals with marginalization.

14.2 Symbolic computation

In this section, we give an overview of the principal techniques to simplify the calculation needed either to evaluate $\mathbf{P}(Searched \mid known)$, or to find the most probable value for *Searched*, or to draw values for *Searched* according to this distribution.

The goal is to perform symbolic computation on the expression

$$\frac{1}{\Sigma} \times \sum_{Free} \left[\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i) \right] \tag{14.6}$$

to obtain another expression to compute the same result with far fewer elementary operations (sum and product). It is called symbolic computation because this can be done independently of the possible numerical values of the considered variables.

We will present these different algorithms as pure and simple algebraic manipulations of Expression (14.6) above, even if most of them have been historically proposed from different points of view (especially in the form of manipulation of graphs and message passing along their arcs).

14.2.1 Exact symbolic computation

We first restrict our analysis to mathematically exact symbolic computation that lead to a simplified expression mathematically equivalent to the starting one.

Question-specific symbolic computation

It is seldom possible to solve analytically the question $\mathbf{P}(Searched \mid known)$. Most of the time, the integral in Expression (14.6) has no explicit solution.

However, this is possible for Kalman filters as defined by the Bayesian program

(13.9). This explains their popularity and their importance in applications. Indeed, once analytically solved, the answer to the question may be computed very efficiently.

Question-dependent symbolic computation

We first take an example to introduce the different possible simplifications.

This example is defined by the following decomposition (14.7) of a joint distribution of nine variables:

$$\begin{aligned}&\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_9)\\ &= \mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(X_4 \mid X_2) \times \mathbf{P}(X_5 \mid X_2)\\ &\times \mathbf{P}(X_6 \mid X_3) \times \mathbf{P}(X_7 \mid X_3) \times \mathbf{P}(X_8 \mid X_6) \times \mathbf{P}(X_9 \mid X_6)\end{aligned} \tag{14.7}$$

corresponding to the Bayesian net defined in Figure 14.1:

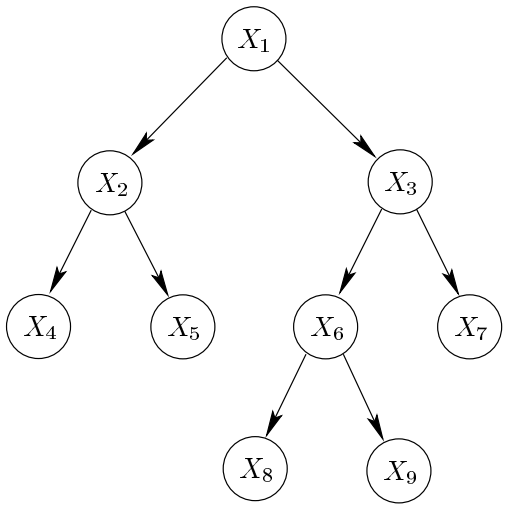

Figure 14.1: The Bayesian network corresponding to the joint distribution in (14.7).

Take, for instance, the question $\mathbf{P}(X_1 \mid x_5 \wedge x_7)$ where $Searched = X_1$, $known = x_5 \wedge x_7$, and $Free = X_2 \wedge X_3 \wedge X_4 \wedge X_6 \wedge X_8 \wedge X_9$. We know that:

$$\mathbf{P}(X_1 \mid x_5 \wedge x_7)$$

$$= \sum_{\substack{X_2 \wedge X_3 \wedge X_4 \\ X_6 \wedge X_8 \wedge X_9}} \left[\begin{array}{c} \mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(X_4 \mid X_2) \times \mathbf{P}(x_5 \mid X_2) \\ \times \mathbf{P}(X_6 \mid X_3) \times \mathbf{P}(x_7 \mid X_3) \times \mathbf{P}(X_8 \mid X_6) \times \mathbf{P}(X_9 \mid X_6) \end{array} \right] \tag{14.8}$$

If each of the variables X_i may take 10 different possible values, then evaluating Expression (14.8) for a given value of X_1 requires 9×10^6 elementary operations.

To reduce this number, we can first reorder the different summations the following way:

$$\mathbf{P}(X_1 \mid x_5 \wedge x_7)$$

$$= \sum_{X_2 \wedge X_3} \left[\begin{array}{c} \mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_5 \mid X_2) \times \mathbf{P}(x_7 \mid X_3) \\ \sum_{X_4} \left[\mathbf{P}(X_4 \mid X_2) \times \sum_{X_6} \left[\mathbf{P}(X_6 \mid X_3) \times \sum_{X_8} \left[\mathbf{P}(X_8 \mid X_6) \times \sum_{X_9} [\mathbf{P}(X_9 \mid X_6)] \right] \right] \right] \end{array} \right] \tag{14.9}$$

and we see that $\sum_{X_9} [\mathbf{P}(X_9 \mid X_6)]$ vanishes as it sums to one. We obtain:

$$\mathbf{P}(X_1 \mid x_5 \wedge x_7)$$

$$= \sum_{X_2 \wedge X_3} \left[\begin{array}{c} \mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_5 \mid X_2) \times \mathbf{P}(x_7 \mid X_3) \\ \sum_{X_4} \left[\mathbf{P}(X_4 \mid X_2) \times \sum_{X_6} \left[\mathbf{P}(X_6 \mid X_3) \times \sum_{X_8} [\mathbf{P}(X_8 \mid X_6)] \right] \right] \end{array} \right] \tag{14.10}$$

This same simplification can also be applied to the sums on X_8, X_6, and X_4 to yield:

$$\mathbf{P}(X_1 \mid X_5 \wedge X_7)$$

$$= \sum_{X_2 \wedge X_3} [\mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_5 \mid X_2) \times \mathbf{P}(x_7 \mid X_3)] \tag{14.11}$$

Evaluating Expression (14.11) requires 5×10^2 elementary operations. Eliminating the parts of the global sum that sum to one is indeed a very efficient simplification.

However, rearranging further the order of the sums in (14.11) may lead to more gains. First, $\mathbf{P}(X_1)$ may be factorized out of the sum:

$$\begin{aligned}&\mathbf{P}(X_1 \mid x_5 \wedge x_7)\\ &= \mathbf{P}(X_1) \times \sum_{X_2 \wedge X_3} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_5 \mid X_2) \times \mathbf{P}(x_7 \mid X_3)]\end{aligned} \tag{14.12}$$

Then, the sum on X_2 and X_3 can be split, leading to:

$$\begin{aligned}&\mathbf{P}(X_1 \mid x_5 \wedge x_7)\\ &= \mathbf{P}(X_1) \times \sum_{X_2} \Big[\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2) \times \sum_{X_3} [\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)]\Big]\end{aligned} \tag{14.13}$$

and as $\sum_{X_3}[\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)]$ is only a function of X_1, it can be factored out of the sum on X_2, to finally have:

$$\begin{aligned}&\mathbf{P}(X_1 \mid x_5 \wedge x_7)\\ &= \mathbf{P}(X_1) \times \sum_{X_3} [\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)] \times \sum_{X_2} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)]\end{aligned} \tag{14.14}$$

Evaluating Expression (14.14) requires $19 + 19 + 2 = 40$ elementary operations, five orders of magnitude less than that required for the initial expression to perform the exact same calculation!

Other questions lead to similar symbolic simplifications. For instance, for the question $\mathbf{P}(X_2 \mid x_5 \wedge x_7)$, we have:

$$\begin{aligned}&\mathbf{P}(X_2 \mid x_5 \wedge x_7)\\ &= \sum_{\substack{X_1 \wedge X_3 \wedge X_4\\ X_6 \wedge X_8 \wedge X_9}} \Big[\begin{array}{c}\mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(X_4 \mid X_2) \times \mathbf{P}(x_5 \mid X_2)\\ \times \mathbf{P}(X_6 \mid X_3) \times \mathbf{P}(x_7 \mid X_3) \times \mathbf{P}(X_8 \mid X_6) \times \mathbf{P}(X_9 \mid X_6)\end{array}\Big]\end{aligned} \tag{14.15}$$

which can be simplified into:

$$\begin{aligned}&\mathbf{P}(X_2 \mid x_5 \wedge x_7)\\ &= \mathbf{P}(x_5 \mid X_2) \times \sum_{X_1}\left[\mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \sum_{X_3}\left[\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)\right]\right]\end{aligned} \tag{14.16}$$

However, the final simplification step is not similar to the simplification of $\mathbf{P}(X_1 \mid x_5 \wedge x_7)$ because $\sum_{X_3}\left[\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)\right]$ depends on X_1 and consequently cannot be factored out of the sum on X_1.

Evaluating expression (14.16) requires $(21 \times 10) + 9 + 1 = 220$ elementary operations, compared with the 9×10^6 necessary for Expression (14.15).

To perform these symbolic simplifications, it is only necessary to apply two fundamental rules: the distributive law and the normalization rule (12.9).

Simplification of

$$\mathbf{P}(Searched \mid known) = \frac{1}{\Sigma} \times \sum_{Free}\left[\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i)\right] \tag{14.17}$$

may be done in three steps:

1. *Eliminate distributions that sum to one*: When a term $\mathbf{P}(L_i \mid R_i)$ appears in the sum, if all the variables appearing in L_i are summed and none of them appears in any of the other R_j, then $\mathbf{P}(L_i \mid R_i)$ sums to one and vanishes out of the global sum. Of course, the list of summed variables, initialized to $Free$, must then be updated by removing the variables of L_i. This process can be recursively applied until no more terms of the product can be removed. It leads to an expression of the form:

$$\mathbf{P}(Searched \mid known) = \frac{1}{\Sigma} \times \sum_{Summed}\left[\prod_j \mathbf{P}(L_j \mid R_j)\right] \tag{14.18}$$

where $Summed \subseteq Free$. An example of this was given in Equations (14.9) and (14.10).

2. *Factorize:* Each term of the remaining product $\prod_j \mathbf{P}(L_j \mid R_j)$, where all the vari-

ables are either *Searched* or *Known* is independent of the variables appearing in *Summed*, and consequently it can be factored out of the sum. We then obtain a new expression of the form:

$$\mathbf{P}(Searched \mid known) = \frac{1}{\Sigma} \times \prod_{k} \mathbf{P}(L_k \mid R_k) \times \sum_{Summed} \prod_{l} \mathbf{P}(L_l \mid R_l) \qquad (14.19)$$

An example of this factorization was given in Equation (14.12).

3. *Order the sums cleverly*: Finally, the last type of simplification that can be made is to reorder the sums of $\sum_{Summed} \prod_{l} \mathbf{P}(L_l \mid R_l)$, to minimize the number of operations required. This third step is much more complicated than the two previous ones: finding the optimal ordering is indeed NP-hard (Arnborg et al. 87). Only heuristics can be proposed but they are useful even if they do not find the optimal ordering. Any ordering helps to break the exponential complexity of the computation of the sum.

Numerous algorithms have been proposed to deal with these simplifications. Among the most interesting or most well known are the SPI[1] algorithm (Shachter, D'Ambrosio & Del Favero, 1990; Li & D'Ambrosio, 1994), the variable elimination family (Zhang & Poole, 1994; Zhang & Poole, 1996), the bucket elimination family of algorithms (Dechter, 1996, Dechter & Rish, 1997), the Query-DAG framework (Darwiche & Provan, 1997), the general distributive law algorithm (Aji & McEliece, 2000) and the SRA[2] algorithm ($$$Mekhnacha2005$$$).

Question-independent symbolic computation

Instead of trying to simplify only:

$$\mathbf{P}(Searched \mid known) = \frac{1}{\Sigma} \times \sum_{Free} \left[\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i) \right] \qquad (14.20)$$

we can try to simplify a family of such questions.

1. Symbolic Probabilistic Inference
2. Successive Restriction Algorithm

For instance, in Bayesian nets, where all the L_i of the decomposition are restricted to a single variable (13.1) it may be interesting to apply symbolic computation to minimize globally the number of numerical operations required for the family of questions:

$$\left\{\begin{array}{l}\mathbf{P}(X_1 \mid known)\\ \mathbf{P}(X_2 \mid known)\\ \quad\ldots\\ \mathbf{P}(X_N \mid known)\end{array}\right\} \tag{14.21}$$

Each of these questions is called a *belief*. The given value of *known* is called the *evidence*.

We return to the example of the previous section. The family of interesting questions is:

$$\left\{\begin{array}{l}\mathbf{P}(X_1 \mid x_5 \wedge x_7)\\ \mathbf{P}(X_2 \mid x_5 \wedge x_7)\\ \mathbf{P}(X_3 \mid x_5 \wedge x_7)\\ \mathbf{P}(X_4 \mid x_5 \wedge x_7)\\ \mathbf{P}(X_6 \mid x_5 \wedge x_7)\\ \mathbf{P}(X_8 \mid x_5 \wedge x_7)\\ \mathbf{P}(X_9 \mid x_5 \wedge x_7)\end{array}\right\} \tag{14.22}$$

Using the simplification scheme of the previous section for each of these seven questions, we obtain:

$$\mathbf{P}(X_1 \mid x_5 \wedge x_7)$$

$$= \mathbf{P}(X_1) \times \sum_{X_3} [\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)] \times \sum_{X_2} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)] \tag{14.23}$$

$$\mathbf{P}(X_2 \mid x_5 \wedge x_7)$$

$$= \mathbf{P}(x_5 \mid X_2) \times \sum_{X_1} \left[\mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \sum_{X_3} [\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)] \right] \tag{14.24}$$

$$\mathbf{P}(X_3 \mid x_5 \wedge x_7)$$

$$= \mathbf{P}(x_7 \mid X_3) \times \sum_{X_1} \left[\mathbf{P}(X_1) \times \mathbf{P}(X_3 \mid X_1) \times \sum_{X_2} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)] \right] \tag{14.25}$$

$$\mathbf{P}(X_4 \mid x_5 \wedge x_7)$$

$$= \sum_{X_2} \left[\begin{array}{c} \mathbf{P}(X_4 \mid X_2) \\ \times \mathbf{P}(x_5 \mid X_2) \times \sum_{X_1} \left[\mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \sum_{X_3} [\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)] \right] \end{array} \right] \tag{14.26}$$

$$\mathbf{P}(X_6 \mid x_5 \wedge x_7)$$

$$= \sum_{X_3} \left[\begin{array}{c} \mathbf{P}(X_6 \mid X_3) \\ \times \mathbf{P}(x_7 \mid X_3) \times \sum_{X_1} \left[\mathbf{P}(X_1) \times \mathbf{P}(X_3 \mid X_1) \times \sum_{X_2} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)] \right] \end{array} \right] \tag{14.27}$$

$$\mathbf{P}(X_8 \mid x_5 \wedge x_7)$$

$$= \sum_{X_6} \left[\begin{array}{c} \mathbf{P}(X_8 \mid X_6) \times \mathbf{P}(X_6 \mid X_3) \\ \times \mathbf{P}(x_7 \mid X_3) \times \sum_{X_1} \left[\mathbf{P}(X_1) \times \mathbf{P}(X_3 \mid X_1) \times \sum_{X_2} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)] \right] \end{array} \right] \tag{14.28}$$

$$\mathbf{P}(X_9 \mid x_5 \wedge x_7)$$

$$= \sum_{X_6} \left[\begin{array}{c} \mathbf{P}(X_9 \mid X_6) \times \mathbf{P}(X_6 \mid X_3) \\ \times \mathbf{P}(x_7 \mid X_3) \times \sum_{X_1} \left[\mathbf{P}(X_1) \times \mathbf{P}(X_3 \mid X_1) \times \sum_{X_2} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)] \right] \end{array} \right] \tag{14.29}$$

These 7 expression shares many terms. To minimize the number of elementary numerical operations, they should not be computed several times but only once. This implies an obvious order in these computations:

- *Step 0*: First, $\mathbf{P}(x_5 \mid X_2)$ and $\mathbf{P}(x_7 \mid X_3)$, which appear everywhere and can be computed immediately.

- *Step 1*: Then $\sum_{X_2}[\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)]$ and $\sum_{X_3}[\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)]$ can be computed directly.

- *Step 2*: In the third step, the first belief $\mathbf{P}(X_1 \mid x_5 \wedge x_7)$ can be evaluated:

$$\begin{aligned}&\mathbf{P}(X_1 \mid x_5 \wedge x_7)\\ &= \mathbf{P}(X_1) \times \sum_{X_3}[\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)] \times \sum_{X_2}[\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)]\end{aligned} \tag{14.30}$$

- *Step 3*: Then the two questions $\mathbf{P}(X_2 \mid x_5 \wedge x_7)$ and $\mathbf{P}(X_3 \mid x_5 \wedge x_7)$ can be solved:

$$\begin{aligned}&\mathbf{P}(X_2 \mid x_5 \wedge x_7)\\ &= \mathbf{P}(x_5 \mid X_2) \times \sum_{X_1}\left[\mathbf{P}(X_2 \mid X_1) \times \frac{\mathbf{P}(X_1 \mid x_5 \wedge x_7)}{\sum_{X_2}[\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(x_5 \mid X_2)]}\right]\end{aligned} \tag{14.31}$$

$$\begin{aligned}&\mathbf{P}(X_3 \mid x_5 \wedge x_7)\\ &= \mathbf{P}(x_7 \mid X_3) \times \sum_{X_1}\left[\mathbf{P}(X_3 \mid X_1) \times \frac{\mathbf{P}(X_1 \mid x_5 \wedge x_7)}{\sum_{X_3}[\mathbf{P}(X_3 \mid X_1) \times \mathbf{P}(x_7 \mid X_3)]}\right]\end{aligned} \tag{14.32}$$

- *Step 4*: The next two expressions $\mathbf{P}(X_4 \mid x_5 \wedge x_7)$ and $\mathbf{P}(X_6 \mid x_5 \wedge x_7)$, can be deduced directly from the two previous ones as:

$$\mathbf{P}(X_4 \mid x_5 \wedge x_7)$$

$$= \sum_{X_2} [\mathbf{P}(X_4 \mid X_2) \times \mathbf{P}(X_2 \mid x_5 \wedge x_7)] \tag{14.33}$$

$$\mathbf{P}(X_6 \mid x_5 \wedge x_7)$$

$$= \sum_{X_3} [\mathbf{P}(X_6 \mid X_3) \times \mathbf{P}(X_3 \mid x_5 \wedge x_7)] \tag{14.34}$$

- *Step 5*: Finally, the last two questions can be computed:

$$\mathbf{P}(X_8 \mid x_5 \wedge x_7)$$

$$= \sum_{X_6} [\mathbf{P}(X_8 \mid X_6) \times \mathbf{P}(X_6 \mid x_5 \wedge x_7)] \tag{14.35}$$

$$\mathbf{P}(X_9 \mid x_5 \wedge x_7)$$

$$= \sum_{X_6} [\mathbf{P}(X_9 \mid X_6) \times \mathbf{P}(X_6 \mid x_5 \wedge x_7)] \tag{14.36}$$

This order of computation may be interpreted as a message-passing algorithm in the Bayesian net (see Figure 14.2 below).

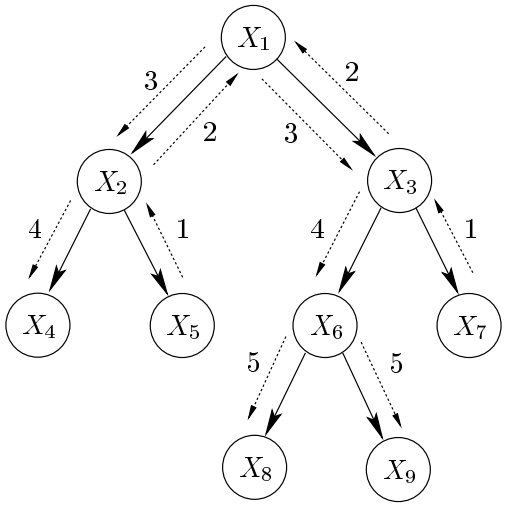

Figure 14.2: The order of computation described by steps 1 to 6 (above) may be interpreted as a message-passing algorithm in the Bayesian net.

This algorithm was simultaneously and independently proposed by Judea Pearl ($$$Pearl86$$$ and Pearl, 1988) under the name of *Belief Propagation,* and by Lauritzen and Spiegelhalter (Spiegelhalter, 1986; Lauritzen & Spiegehalter, 1988; Lauritzen, 1996) as the *Sum-Product* algorithm.

When the graph associated with the Bayesian network has no undirected cycles[1], it is always possible to find this ordering, ensuring that each sub-expression is evaluated once and only once.

On the other hand, when the graph of the Bayesian net has some undirected cycles the situation is trickier and such a clever ordering of the computation may not be found.

For instance, let us modify the above example by adding a dependency between X_2 and X_3. We then obtain the new decomposition:

$$\begin{aligned}
&\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_9) \\
&= \mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_2 \wedge X_1) \times \mathbf{P}(X_4 \mid X_2) \times \mathbf{P}(X_5 \mid X_2) \\
&\times \mathbf{P}(X_6 \mid X_3) \times \mathbf{P}(X_7 \mid X_3) \times \mathbf{P}(X_8 \mid X_6) \times \mathbf{P}(X_9 \mid X_6)
\end{aligned} \tag{14.37}$$

which corresponds to the graph of Figure 14.3 below.

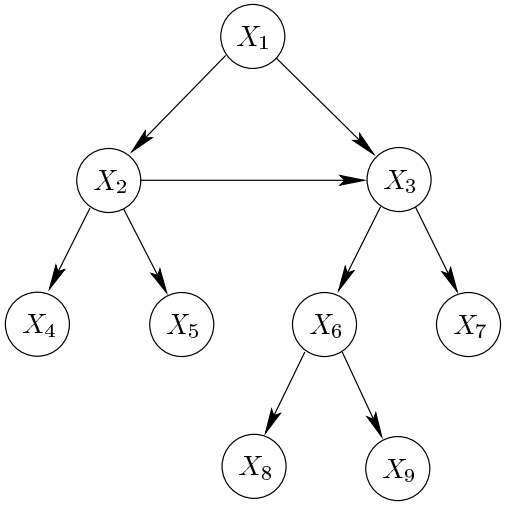

Figure 14.3: The Bayesian network corresponding to the joint distribution in (14.37).

Applying the simplification rules to the different questions, we obtain:

1. it is either a tree or a polytree

$$\mathbf{P}(X_1 \mid x_5 \wedge x_7)$$

$$= \mathbf{P}(X_1) \times \sum_{X_2 \wedge X_3} [\mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_2 \wedge X_1) \times \mathbf{P}(x_5 \mid X_2) \times \mathbf{P}(x_7 \mid X_3)] \tag{14.38}$$

$$\mathbf{P}(X_2 \mid x_5 \wedge x_7)$$

$$= \mathbf{P}(x_5 \mid X_2) \times \sum_{X_1 \wedge X_3} [\mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_2 \wedge X_1) \times \mathbf{P}(x_7 \mid X_3)] \tag{14.39}$$

$$\mathbf{P}(X_3 \mid x_5 \wedge x_7)$$

$$= \mathbf{P}(x_7 \mid X_3) \times \sum_{X_1 \wedge X_2} [\mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1) \times \mathbf{P}(X_3 \mid X_2 \wedge X_1) \times \mathbf{P}(x_5 \mid X_2)] \tag{14.40}$$

The four other cases are unchanged relative to these three (see (14.33), (14.34), (14.35) and (14.36)).

Obviously, the different elements appearing in these three expressions may not be neatly separated as in the previous case. The conjunction of variables $X_1 \wedge X_2 \wedge X_3$ must be considered as a whole: they form a new variable $A = X_1 \wedge X_2 \wedge X_3$. The decomposition (14.37) becomes:

$$\begin{aligned} &\mathbf{P}(X_1 \wedge X_2 \wedge \ldots \wedge X_9) \\ &= \mathbf{P}(A) \times \mathbf{P}(X_4 \mid A) \times \mathbf{P}(X_5 \mid A) \\ &\times \mathbf{P}(X_6 \mid A) \times \mathbf{P}(X_7 \mid A) \times \mathbf{P}(X_8 \mid X_6) \times \mathbf{P}(X_9 \mid X_6) \end{aligned} \tag{14.41}$$

This corresponds to the graph in Figure 14.4 below, which has a tree structure:

We have recreated the previous case, where the message-passing algorithms may be applied. However, this has not eliminated our troubles completely, because to compute $\mathbf{P}(X_1 \mid x_5 \wedge x_7)$, $\mathbf{P}(X_2 \mid x_5 \wedge x_7)$, and $\mathbf{P}(X_3 \mid x_5 \wedge x_7)$, we shall now require marginalization of the distribution $\mathbf{P}(A \mid x_5 \wedge x_7)$:

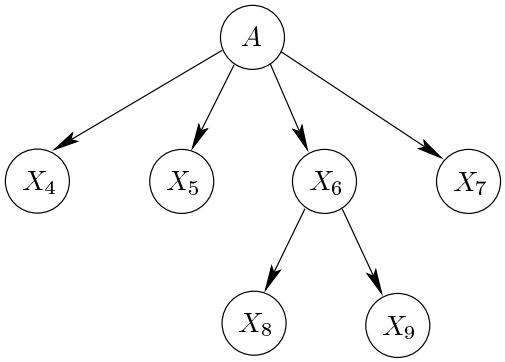

Figure 14.4: The Bayesian network resulting from the introdction of a new variable $A = X_1 \wedge X_2 \wedge X_3$ in the Bayesian network shown in Figure 14.3.

$$\mathbf{P}(X_1 \mid x_5 \wedge x_7) = \sum_{X_2 \wedge X_3} \mathbf{P}(A \mid x_5 \wedge x_7) \tag{14.42}$$

$$\mathbf{P}(X_2 \mid x_5 \wedge x_7) = \sum_{X_1 \wedge X_3} \mathbf{P}(A \mid x_5 \wedge x_7) \tag{14.43}$$

$$\mathbf{P}(X_3 \mid x_5 \wedge x_7) = \sum_{X_1 \wedge X_2} \mathbf{P}(A \mid x_5 \wedge x_7) \tag{14.44}$$

Indeed, the computation of these sums may be very expensive.

The *junction tree* algorithm, also often called *JLO* after its inventors (see Jensen, Lauritzen & Olesen, 1990), searches for such a decomposition and implements the corresponding computation to solve $\mathbf{P}(X_i \mid Known)$.

The main idea of the junction tree algorithm is to convert the Bayesian network into a tree by clustering the nodes together. After building this tree of clusters, inference can be done efficiently by the single message-passing algorithm.

All the details of this popular algorithm may be found in the textbooks concerning graphical models cited previously (see Section 13.1.1) and also in Jensen's book (Jensen, 1996).

It is very important to note that this JLO algorithm may lead to very poor simplification, as, in some cases, the required marginalizations may be very expensive. The cost of these marginalizations grows exponentially with the number of variables in the conjunctions being considered.

Historically, another solution was proposed by Pearl (Pearl, 1988) to deal with graph with undirected cycles. It is called the *cut-set* algorithm. The principle of the cut-set algorithm is to break the cycles by fixing the values of some of the variables in these cycles and computing in turn the different questions for these different possible values.

When there are no *Free* variables and if we are interested in the most probable value of *Searched*, the usual equation:

$$\mathbf{P}(Searched \mid known) = \frac{1}{\Sigma} \times \sum_{Free} \left[\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i) \right] \tag{14.45}$$

is transformed into:

$$\mathbf{MAX}_{Searched}(\mathbf{P}(Searched \mid known)) = \frac{1}{\Sigma} \times \mathbf{MAX}_{Searched} \left[\mathbf{P}(L_1) \times \prod_{i=2}^{\kappa} \mathbf{P}(L_i \mid R_i) \right] \tag{14.46}$$

The distributive law applies to the couple ($\mathbf{MAX}$, $\prod$) in the same way as it applies to the couple ($\sum$, $\prod$). Consequently, most of the previous simplifications are still valid with this new couple of operator.

The *sum-product* algorithm becomes the *max-product* algorithm, or more commonly, the *min-sum* algorithm, as it may be further transformed by operating on the inverse of the logarithm.

It is also known as the *Viterbi* algorithm and it is particularly used with HMMs. to find the most probable series of states that leads to the present state, knowing the past observations as stated in Bayesian program (13.8). In that case, we have:

$$Searched = S^1 \wedge S^2 \wedge \ldots \wedge S^{t-1} \tag{14.47}$$

$$known = s^t \wedge o^0 \wedge \ldots \wedge o^t \tag{14.48}$$

$$Free = \varnothing \tag{14.49}$$

and the question is:

$$\mathbf{MAX}_{S^1 \wedge S^2 \wedge \ldots \wedge S^{t-1}}[\mathbf{P}(S^1 \wedge S^2 \wedge \ldots \wedge S^{t-1} \mid s^t \wedge o^0 \wedge \ldots \wedge o^t)] \tag{14.50}$$

14.2.2 Approxiamate symbolic computation

Often, the exact symbolic simplification methods described previously are not sufficient. We must then use approximate symbolic simplifications that lead to mathematical expressions that approach the values of $\mathbf{P}(Searched \mid known)$.

Variational methods

The key to *variational methods* is to convert a probabilistic inference problem into an optimization problem. In this way, the standard tools of constrained optimization can be used to solve the inference problem. The idea is to replace a joint distribution $\mathbf{P}(X) = \mathbf{P}(X_1 \wedge \ldots \wedge X_N)$ (represented by an acyclic graph in the case of a Bayesian net) by an approximation $\mathbf{Q}(X)$, and to compute the Kullback-Leibler divergence between the two distributions.

To do that, we consider the energy of a configuration X defined by:

$$\mathbf{E}(X) = -\mathbf{log}(\mathbf{P}(X)) - \mathbf{log}(Z) \tag{14.51}$$

and the variational free energy (or Kullback-Leibler distance) as:

$$\mathbf{F}(\mathbf{Q}, \mathbf{P}) = \sum_X \left[\mathbf{Q}(X) \times \mathbf{log}\left(\frac{\mathbf{Q}(X)}{\mathbf{P}(X)}\right)\right] - \mathbf{log}(Z) \tag{14.52}$$

This distance is minimized when $\mathbf{P}(X) = \mathbf{Q}(X)$.

Of course, minimizing $\mathbf{F}$ is as difficult as the original inference problem. However, by considering a different family of $\mathbf{Q}(X)$, we obtain a different approximation of $\mathbf{F}$ and as a consequence, different variational methods.

For example, if one restricts oneself to the family of factorized independent distributions:

$$\mathbf{Q}(X) = \prod_{i=1}^{N} \mathbf{Q}_i(X_i) \tag{14.53}$$

the variational method boils down to the *mean field* approximation. Minimizing $\mathbf{F}(\mathbf{Q}, \mathbf{P})$ is greatly simplified using the acyclic graph structure of $\mathbf{P}(X)$.

A general introduction to variational methods may be found in two introductory texts by Jordan (Jordan et al., 1999; Jordan & Weiss, 2002). Interesting refinements are described in Yedidia's paper (Yedidia, Freeman & Weiss, 2002).

Helmotz Machine

$$$more to come$$$

14.3 Numerical computation

Once the symbolic simplifications are complete, we must finally do the numerical computation of the simplified expression.

14.3.1 Searching the modes in high-dimensional spaces

The central problem to solve is to try to find the mode of the simplified expression in a very high-dimensional space. This is indeed a classical optimization problem, where classical optimization methods can be applied.

However, we must choose to either sample points from the expression or to build an explicit, even approximate representation of the distribution that can be saved in memory and used again later.

Distribution sampling methods

Direct sampling methods

In some cases, it is possible, when disposing of a uniform random generator, to use a *transformation function* to sample a given distribution $\mathbf{P}(X)$ (see Rubinstein, 1981).

One of the more important transformation functions is the Box-Muller transformation (Box & Muller, 1958). This transformation allows us to generate a set of random numbers from a one-dimensional normal distribution using another set of random numbers generated by a uniform random generator.

Sampling a multi-dimensional normal distribution $\mathbf{P}(X) = \mathbf{G}(X, \mu, \sigma)$ is also possible by diagonalizing the variance matrix Σ. By diagonalization, the major axes (eigenvectors) obtained define a new coordinate system (in which the components are independent). Therefore, the problem is reduced to drawing each component independently from D one-dimensional normal distributions having, for each component, the

eigenvalue v_i as variance. The vector obtained, $x^{Diag} = [x_1^{Diag}, ..., x_D^{Diag}]$, is then rewritten in the original coordinate system by multiplying it by the transpose of the eigenmatrix to obtain the sample vector $x = [x_1, ..., x_D]$.

When disposing (analytically or numerically) of the repartition function $\mathbf{F}(X)$ corresponding to a target distribution $\mathbf{f}(X)$:

$$\mathbf{F}(X) = \int_{-\infty}^{X} \mathbf{f}(i)di \tag{14.54}$$

the problem of sampling $\mathbf{f}(X)$ is reduced to the problem of inverting the repartition function $\mathbf{F}(X)$. Drawing a sample point from $\mathbf{f}(X)$ requires:

1. drawing a uniform random value $u \in [0, 1]$;

2. calculating the drawn point as $x = \mathbf{F}^{-1}(u)$.

In the general case, analytical forms of the repartition function $\mathbf{F}(X)$ and its inverse are not available. Explicitly computing the repartition function $\mathbf{F}(X)$ and inverting it numerically is possible for low-dimensional spaces. It is usually impossible in practice for high-dimensional cases.

Monte Carlo methods

Monte Carlo methods group together several different methods for sampling in high-dimensional spaces. In this section, we present some of the most popular variants: importance sampling, rejection sampling, Metropolis sampling, and Gibbs sampling.

Two excellent starting points on Monte Carlo methods are the tutorials by Neal (Neal, 1993) and MacKay (MacKay, 1996).

Importance sampling

Suppose we are interested in sampling a distribution $\mathbf{P}(X)$ for which no direct sampling method is available and that we are able to evaluate this distribution for each point x_i of the state space.

Suppose also that we have a simpler distribution $\mathbf{Q}(X)$ (called the *proposal distribution*) that we can also evaluate for each point x_i and for which a direct sampling method is available.

Using *importance sampling* to sample $\mathbf{P}(X)$ consists of generating N pairs $\{(x_1, \omega_1), ..., (x_N, \omega_N)\}$ where the points x_i are drawn from $\mathbf{Q}(X)$ and where:

$$\omega_i = \frac{\mathbf{P}(x_i)}{\mathbf{Q}(x_i)} \tag{14.55}$$

This sampling method is especially useful for Monte Carlo importance sampling integration method (see also Section 16.5.2).

Rejection sampling

Suppose we are interested in sampling a distribution $\mathbf{P}(X)$ for which no direct sampling method is available and that we are able to evaluate this distribution for each point x_i of the state space.

Suppose also that we have a simpler distribution $\mathbf{Q}(X)$ that we can evaluate for each point x_i, for which a direct sampling method is available, and that the distribution respects the constraint:

$$\exists c, \forall x, [c \times \mathbf{Q}(x) > \mathbf{P}(x)] \tag{14.56}$$

Using *rejection sampling* to draw a point of $\mathbf{P}(X)$ consists of drawing a point x_i from $\mathbf{Q}(X)$ and accepting it with a probability of:

$$\frac{c \times \mathbf{Q}(x_i)}{\mathbf{P}(x_i)} \tag{14.57}$$

The complete procedure is then:

1. Draw a candidate point x_i from $\mathbf{Q}(X)$;

2. Evaluate $c \times \mathbf{Q}(x_i)$;

3. Generate a uniform random value $u \in [0, c \times \mathbf{Q}(x_i)]$;

4. Either accept the point x_i if $\mathbf{P}(x_i) > u$, or reject this point and draw another one.

It is clear that this rejection sampling will be efficient if the distribution $\mathbf{Q}(X)$ is a

good approximation of $\mathbf{P}(X)$. Otherwise, the rejection rate will be very important.

Metropolis sampling

Previous methods using a *proposal distribution* perform well if $\mathbf{Q}(X)$ is a good approximation of $\mathbf{P}(X)$. For complex problems in high-dimensional spaces, it is difficult to find such a distribution.

The *Metropolis algorithm* (Metropolis et al., 1953) uses a Markovian process in which a sequence of states x^t is generated. The new state x^t depends on the previous one x^{t-1}. This algorithm is an example of *Markov Chain Monte Carlo* (MCMC) methods.

Instead of using a single proposal distribution $\mathbf{Q}(X)$, the Metropolis algorithm uses a proposal distribution $\mathbf{Q}(X_i, X^t)$, which depends on the current state x^t. This distribution can be a simple distribution (a normal distribution having x^t as the mean value, for example).

Suppose the current state is x^t. A candidate x_i is generated from $\mathbf{Q}(X_i, X^t)$. To accept or reject this candidate we must compute:

$$a = \frac{\mathbf{P}(x_i) \times \mathbf{Q}(x_i, x^t)}{\mathbf{P}(x^t) \times \mathbf{Q}(x^t, x_i)} \tag{14.58}$$

If $a > 1$ then x_i is accepted. Otherwise, it is accepted with probability a. If x_i is accepted, we set $x^{t+1} = x_i$. If x_i is rejected, then we set $x^{t+1} = x^t$.

A frequent choice for the proposal distribution $\mathbf{Q}(X_i, X^t)$ is to choose a symmetrical distribution (a normal distribution having x^t as the mean value, for example). Then:

$$\frac{\mathbf{Q}(x_i, x^t)}{\mathbf{Q}(x^t, x_i)} = 1 \tag{14.59}$$

and we obtain:

$$a = \frac{\mathbf{P}(x_i)}{\mathbf{P}(x^t)} \tag{14.60}$$

One drawback of MCMC techniques is that we must generally wait for the chain to reach equilibrium. This can take a long time, and it is sometimes difficult to tell when it happens.

Gibbs sampling

Gibbs sampling, also known as the *heatbath method*, is another example of a Markov Chain Monte Carlo sampling technique. It has come into prominence only recently with the works of Geman and Smith (Geman & Geman, 1984 and Smith & Roberts, 1993). It is a method for sampling from distributions over at least two dimensions, and can be viewed as a Metropolis method in which the proposal distribution $\mathbf{Q}(X)$ is defined in terms of the conditional distributions of the joint distribution $\mathbf{P}(X)$. It is assumed that while $\mathbf{P}(X)$ is too complex to draw samples from directly, its conditional distributions $\mathbf{P}(X_i \mid X_1 \wedge ... \wedge X_{i-1} \wedge X_{i+1} \wedge ... \wedge X_N)$ are tractable.

In the general case of a system of N variables, a single iteration involves sampling one variable at a time:

$$\begin{aligned} x_1^{t+1} &\sim \mathbf{P}(X_1 \mid x_2^t \wedge ... \wedge x_N^t) \\ x_2^{t+1} &\sim \mathbf{P}(X_2 \mid x_1^t \wedge x_3^t \wedge ... \wedge x_N^t) \\ &... \\ x_N^{t+1} &\sim \mathbf{P}(X_N \mid x_1^t \wedge ... \wedge x_{N-1}^t) \end{aligned} \tag{14.61}$$

Building an explicit representation of the distribution

Representation using a sample set of points

Suppose we are interested in building an explicit representation of a given D-dimensional distribution $\mathbf{P}(X)$.

The simplest representation one can imagine is to encode $\mathbf{P}(X)$ as a sample set of N points:

$$\mathbf{P}(X) \equiv \{x_1, ..., x_N\} \tag{14.62}$$

This kind of representation, although simple, can be very useful if this distribution is used in a sampling process *directly* or *indirectly*.

We say that $\mathbf{P}(X)$ is used *directly* in a sampling process if the problem consists of drawing a set of M points $\{x_1, ..., x_N\}$.

We say that $\mathbf{P}(X)$ is used *indirectly* in a sampling process if the problem consists of

estimating a distribution $\mathbf{P}(Y)$ as an integral on the X space using a Monte Carlo method:

$$\mathbf{P}(Y) = \sum_X \mathbf{P}(X) \times \mathbf{P}(Y \mid X) \tag{14.63}$$

Drawing a point directly from $\mathbf{P}(X)$ consists of drawing uniformly an integer value i between one and N, and returning the corresponding point in the set $\{x_1, ..., x_N\}$. This process may be iterated M times to obtain M points.

If $\mathbf{P}(X)$ is used indirectly, then $\mathbf{P}(Y)$ may be estimated using a Monte Carlo integration method (see Section 14.3.2):

$$\mathbf{P}(Y) \approx \frac{1}{N} \times \sum_{i=1}^{N} \mathbf{P}(Y \mid [X = x_i]) \tag{14.64}$$

This kind of representation can easily be improved by encoding $\mathbf{P}(X)$ as a sample set of N couples instead of N points:

$$\mathbf{P}(X) \equiv \{(x_1, \omega_1), ..., (x_N, \omega_N)\} \tag{14.65}$$

where the ω_i are weight values, proportional to $\mathbf{P}(x_i)$.

This kind of representation is especially used for particle filters (see Section 13.1.2). When $\mathbf{P}(X)$ is used for estimating a sum, we obtain:

$$\mathbf{P}(Y) \approx \frac{\sum_{i=1}^{N} [\omega_i \times \mathbf{P}(Y \mid [X = x_i])]}{\sum_{i=1}^{N} \omega_i} \tag{14.66}$$

Both kinds of representation clearly suffer from important limitations:

- It is obviously very difficult to represent huge high-dimensional spaces with a reasonable number of points.
- It is very difficult to place the points at the modes where they are significant.
- It is clear that these methods of representation are unable to make generalization in

the neighborhood of the points of the sample set $\mathbf{P}(X)$, and can only be evaluated for these points.

Multi-Resolution Binary Tree

Unlike the preceding representations, a Multi-Resolution Binary Tree (MRBT) is an explicit representation having a capacity for generalization. Using an MRBT, we can compute, for all points x, the probability value $\mathbf{P}([X = x])$.

The main idea of MRBTs is that high-probability regions of the distribution $\mathbf{P}(X)$ should be represented with high resolution, while low-probability regions may be represented with low resolution.

An MRBT is built incrementally by inserting N pairs $\{(x_1, \mathbf{P}([X = x_1])), ..., (x_N, \mathbf{P}([X = x_N]))\}$. This set of pairs is generated using an external process such as a genetic algorithm or a Metropolis sampler.

When inserting a given pair $(x_i, \mathbf{P}([X = x_i]))$, the binary tree is updated so that the resolution of the region including the point x_i is increased by splitting the corresponding node on one dimension.

The built MRBT can be used to find the probability value of any given point x (i.e. to compute $\mathbf{P}([X = x])$) by searching (using dichotomy) for the leaf node corresponding to x and returning its probability value.

To draw a point from the distribution $\mathbf{P}(X)$, we also use dichotomy to draw a leaf node. Starting from the root node, we choose to go to its first or second child according to their respective total probability values. This procedure is iterated until a leaf node is reached. Having this leaf node, drawing a value x from $\mathbf{P}(X)$ consists of a uniform draw in the region encoded by this leaf.

More details on MRBT implementation and use can be found in a patent protecting this method (Bessière, 2002) and also in (Bellot & Bessière, 2003).

14.3.2 Marginalization (integration) in high-dimensional spaces

Integral calculus is the basis of Bayesian inference. Unfortunately, analytic methods for integral evaluation seem very limited in real-world applications, where integrands may have complex shapes and integrations spaces may have very high dimensionality.

Domain subdivision-based methods (such as trapezoidal or Simpson methods) are

deterministic numerical techniques often used for numerical integration in low-dimensional spaces. However, these techniques are poorly adapted for high-dimensional cases. These techniques will not be discussed further but you should be aware of them. Good sources for such techniques are the numerical recipes (Press et al., 1992) and a book by Davis (Davis & Rabinowitz, 1975).

Monte Carlo methods (MC) are powerful stochastic simulation techniques that may be applied to solve optimization and numerical integration problems in large-dimensional spaces. Since their introduction in the physics literature in the 1950s, Monte Carlo methods have been at the center of the recent Bayesian revolution in applied statistics and related fields.

Analytical integration

Analytical solutions to integration problems are available in well-catalogued instances in which the integrands have particular shapes. Gradshteyn's book (Gradshteyn & Ryzhik, 1980) is a useful and standard reference on these methods.

In probabilistic inference, the best-known and most interesting particular case is when the integrand is a product of generalized normals (Dirac delta functions and Gaussians) and when the model is linear or can be linearized (variance matrices are small enough). If we have to compute the integral:

$$I(Y) = \int_{-\infty}^{\infty} \mathbf{P}(X) \times \mathbf{P}(Y \mid X) \times dX^D, \tag{14.67}$$

where:

- $\mathbf{P}(X) = \mathbf{G}(X, \mu_X, \Sigma_X)$
- $\mathbf{P}(Y \mid X) = \mathbf{G}(Y, A \bullet X, \Sigma_Y)$

then we obtain the analytical solution:

$$I(Y) = \mathbf{G}(Y, [A \bullet \mu_X], [A \bullet \Sigma_X \bullet A^T + \Sigma_Y]), \tag{14.68}$$

where A is a constant matrix (or a Jacobian in a linearized model) and A^T is its transpose.

This analytical resolution of the integral is used extensively in, for example, eKalman filters (see Section 13.1.2), and explains their practical success.

Monte Carlo methods for numerical integration

The aim of Monte Carlo methods is to approximate efficiently the D-dimensional integral (where D can be very large):

$$I = \int \mathbf{P}(X) \times \mathbf{Q}(X) \times dX^D \tag{14.69}$$

We assume that X is a D-dimensional vector with real or discrete components, or a combination. We use the symbol $\int ...$ as a generalized integration operator for both real integrals (over real components) and sums (over discrete components).

Assuming that we cannot visit every single location x_i in the state (integration) space, the simplest solution for estimating the integral (14.69) is to uniformly sample the integration space X and then estimate I by $\widehat{I}$:

$$\widehat{I} = \frac{1}{N} \times \sum_{i=1}^{N} \mathbf{P}(x_i) \times \mathbf{Q}(x_i) \tag{14.70}$$

High-dimensional probability distributions are often concentrated on a small region T of the state (integration) space X, known as its *typical set*. For example, if the space X contains a large number of roughly independent variables, then the volume $\lceil T \rceil$ of the region T is given by:

$$\lceil T \rceil \approx 2^{\mathbf{H}(\mathbf{P}(X))} \tag{14.71}$$

where $\mathbf{H}(\mathbf{P}(X))$ is the Shannon entropy (see MacKay, 1996):

$$\mathbf{H}(\mathbf{P}(X)) = -\sum_{X} \mathbf{P}(X) \times \mathbf{log}(\mathbf{P}(X)) \tag{14.72}$$

The number N of points drawn uniformly for the state (integration) space X must be sufficiently large to cover the region T containing most of the probability mass of $\mathbf{P}(x)$:

$$N \propto \frac{\lceil X \rceil}{\lceil T \rceil} \tag{14.73}$$

where $\lceil X \rceil$ is the volume of the state space. This makes the exploration of the state space using uniform sampling very expensive in the general case.

Instead of exploring the integration space uniformly, Monte Carlo methods try to use the information provided by the distribution $\mathbf{P}(X)$ to explore this space more efficiently. The main idea of these techniques is to approximate the integral I by estimating the expectation of the function $\mathbf{Q}(X)$ under the distribution $\mathbf{P}(X)$:

$$I = \int \mathbf{P}(X) \times \mathbf{Q}(X) \times dX^D = \langle \mathbf{Q}(X) \rangle \tag{14.74}$$

Clearly, if we are able to generate a set of points (vectors) $\{x_1, ..., x_N\}$ from $\mathbf{P}(X)$, the expectation of $\widehat{I}$ is I. In addition, as the number of samples N increases, the variance of the estimator $\widehat{I}$ will decrease as σ^2/N where σ^2 is the variance of $\mathbf{Q}$:

$$\sigma^2 = \int \mathbf{P}(X) \times (\mathbf{Q}(X) - \widehat{Q}) \times dx^D \tag{14.75}$$

and $\widehat{Q}$ is the expectation of $\mathbf{Q}$.

This result is an important property of Monte Carlo methods: the accuracy of Monte Carlo estimates is independent of the dimensionality of the integration space.

Neal (Neal, 1993) offers a good survey of Monte Carlo sampling techniques.

Simple (or perfect) Monte Carlo integration

Suppose we are able to obtain a set of samples $\{x_1, ..., x_N\}$ from the distribution $\mathbf{P}(X)$. We can use these samples to find the estimator:

$$\widehat{I} = \frac{1}{N} \times \sum_{i=1}^{N} \mathbf{Q}(x_i) \tag{14.76}$$

The *perfect* Monte Carlo method assumes the capacity to sample the distribution $\mathbf{P}(X)$ efficiently. This is possible when $\mathbf{P}(X)$ is a standard and simple distribution with a direct sampling method, or a product of standard distributions on which direct sampling is possible using the Gibbs algorithm.

For instance, if $\mathbf{P}(X) = \mathbf{P}(X_1 \wedge X_2) = \mathbf{P}(X_1) \times \mathbf{P}(X_2 \mid X_1)$ where $\mathbf{P}(X_1)$ is a uni-

form distribution and $\mathbf{P}(X_2 \mid X_1)$ is a normal distribution having X_1 as mean and a fixed value as variance, then drawing a point $x^t = (x_1^t, x_2^t)$ consists of:

- drawing x_1^t from $\mathbf{P}(X_1)$ (i.e. uniformly on the possible values).
- drawing x_2^t from the normal distribution $\mathbf{P}(X_2 \mid [X_1 = x_1^t])$.

Importance sampling Monte Carlo integration

Suppose now that we are unable to generate sample points directly from the distribution $\mathbf{P}(X)$ but only from a simpler distribution $\mathbf{Q}(X)$ called the *sampling distribution*.

Using importance sampling, a set of N points is generated from $\mathbf{Q}(X)$. If these sample points were generated from $\mathbf{P}(X)$, we could estimate I using Equation (14.76). However, as these points have been generated from $\mathbf{Q}(X)$ and not from $\mathbf{P}(X)$, the values of X for which $\mathbf{Q}(X)$ is greater than $\mathbf{P}(X)$ will be over-represented and the values for which $\mathbf{Q}(X)$ is less than $\mathbf{P}(X)$ will be under-represented. To take into account this problem, *importance sampling* introduces weights:

$$\omega_i = \frac{\mathbf{P}(x_i)}{\mathbf{Q}(x_i)} \tag{14.77}$$

These weights are used to define the importance of each point in the estimator:

$$\widehat{I} = \frac{1}{\sum_{i=1}^{N} \omega_i} \times \sum_{i=1}^{N} [\omega_i \times \mathbf{Q}(x_i)] \tag{14.78}$$

This method of integration is used especially in particle filters.

15

Bayesian Learning Revisited

$$$

$$$

15.1 Problematic

A Bayesian program has the following structure:

$$\mathbf{Ds}\begin{cases} \text{?Preliminary knowledge?}(\pi) \begin{cases} \Lambda = \Lambda_1 \wedge \ldots \wedge \Lambda_N \\ \mathbf{Dc} \\ \mathbf{P}(X_1 \wedge \ldots \wedge X_N \mid \delta \wedge \pi) = \prod_{i=1}^{M} \mathbf{P}(L_i \mid R_i \wedge \delta \wedge \\ \mathbf{Fo} \\ \mathbf{P}(L_i \mid R_i \wedge \delta \wedge \pi) = \mathbf{F}^i_{\lambda_i}(L^i) \end{cases} \\ \text{Data}(\delta) \end{cases} \tag{15.1}$$

In section 3 we showed that most probabilistic models can be specified using this formalism. In section 4 we presented how to automate computation of such programs. At

this point, the reader should be convinced of the interest of such probabilistic models for both solving important engineering problems and for building interesting models of cognition.

However, an important question has still to be answered: where do the Bayesian programs come from? From the engineering point of view this question is: is there a methodology to develop Bayesian programs? From the scientific point of view this question could be translated as: is there any natural process that could explain the apparition of probabilistic models in the brains of living beings? The purpose of this 5th section is to review the different aspects of these questions.

The global question (where do the Bayesian programs come from?) can be divided into sub-questions, which can be answered separately and can be made mathematically precise:

1. How to identify (learn) the value of the free parameters?

2. How to compare different probabilistic models (specifications)?

3. How to find interesting decompositions and associated parametric forms?

4. How to find the pertinent variables to model a phenomenon?

15.1.1 How to identify (learn) the value of the free parameters?

Given some specification π (consisting of the pertinent variables, the decomposition of the joint distribution and the parametric forms), and given some data set δ, one wants to identify the values of the free parameters that best take into account the data.

It is always possible to make the free parameters appear as an explicit variable Λ of the model:

$$\mathbf{Ds}\left\{\begin{array}{l}\text{?Preliminary knowledge?}(\pi)\left\{\begin{array}{l}X = X_1 \wedge ... \wedge X_N\\ \Lambda = \Lambda_1 \wedge ... \wedge \Lambda_M\\ \mathbf{Dc}\\ \mathbf{P}(X_1 \wedge ... \wedge X_N \wedge \Lambda_1 \wedge ... \wedge \Lambda_M \mid \delta \wedge \pi\\ \quad = \mathbf{P}(\Lambda \mid \delta \wedge \pi) \times \prod_{i=2}^{M} \mathbf{P}(L_i \mid R_i \wedge \Lambda_i \wedge\\ \mathbf{Fo}\\ \mathbf{P}(L^i \mid R^i \wedge \Lambda_i \wedge \pi) = \mathbf{F}^i_{\lambda_i}(L^i)\end{array}\right.\end{array}\right. \tag{15.2}$$

- Λ_i represents the free parameters of the i-th form and Λ the set of all these free parameters.

- The goal of the parameters is to sum up what has been learned from the data. Consequently:

  - The parameters depend on the data: $\mathbf{P}(\Lambda \mid \delta \wedge \pi)$

  - Knowing the parameters, the distributions does not depend any more on the data:

$$\mathbf{P}(L_i \mid R_i \wedge \Lambda_i \wedge \delta \wedge \pi) = \mathbf{P}(L_i \mid R_i \wedge \Lambda_i \wedge \pi) \tag{15.3}$$

Therefore, there is a Bayesian formulation of the parameter identification problem:

$$\mathbf{Ds}\left\{\begin{array}{l}\text{?Preliminary knowledge?}(\pi)\left\{\begin{array}{l}\Delta = {}_1X \wedge ... \wedge {}_PX\\ \Lambda = \Lambda_1 \wedge ... \wedge \Lambda_M\\ \mathbf{Dc}\\ \mathbf{P}(\Delta \wedge \Lambda \mid \pi) = \mathbf{P}(\Lambda \mid \pi) \times \mathbf{P}(\Delta \mid \Lambda\\ \mathbf{Fo}\\ \mathbf{P}(\Delta \mid \Lambda \wedge \pi) = \prod_{k=1}^{P} \mathbf{P}({}_kX \mid \Lambda \wedge \pi)\end{array}\right.\\ \text{Data } (\delta)\end{array}\right. \tag{15.4}$$

- δ is a collection of P data ${}_jx$. Each datum is a collection of values for the variables

$X_1 \wedge \ldots \wedge X_N$. For example, a datum ${}_j x$ is given by ${}_j x = [X_1 = {}_j x_1] \wedge \ldots \wedge [X_N = {}_j x_N]$.

- Each datum, knowing the model π and the values of the parameters Λ, are considered to be independent from one another:

$$\mathbf{P}(\Delta \mid \Lambda \wedge \pi) = \prod_{k=1}^{P} \mathbf{P}({}_k X \mid \Lambda \wedge \pi) \tag{15.5}$$

Learning or identifying the free parameters consist in answering the question $\mathbf{P}(\Lambda \mid \delta \wedge \pi)$: What is the most probable value of the parameters knowing a given data set δ? The answer to this question is obtained by:

$$\begin{aligned} \mathbf{P}(\Lambda \mid \delta \wedge \pi) &= \frac{\mathbf{P}(\Lambda \mid \pi) \times \mathbf{P}(\delta \mid \Lambda \wedge \pi)}{\mathbf{P}(\Delta \mid \pi)} \\ &= \frac{1}{\Sigma} \times \mathbf{P}(\Lambda \mid \pi) \times \mathbf{P}(\delta \mid \Lambda \wedge \pi) \\ &= \frac{1}{\Sigma} \times \mathbf{P}(\Lambda \mid \pi) \times \prod_{k=1}^{P} \mathbf{P}({}_k x \mid \Lambda \wedge \pi) \end{aligned} \tag{15.6}$$

Most often, $\mathbf{P}(\Lambda \mid \pi)$ is supposed to be uniform. Maximizing $\mathbf{P}(\Lambda \mid \delta \wedge \pi)$ thus reduces to the maximization of the likelihood $\mathbf{P}(\delta \mid \Lambda \wedge \pi)$. However, one may very well use strong priors on the values of the parameters.

The difficulty of this problem is that the parameter space may be huge and exploring it to find the most probable value may be untractable.

15.1.2 How to compare different probabilistic models (specifications)?

Given two models π_1 and π_2, each one corresponding to a given Bayesian specification, and given some data set δ, we want to choose the model which best fits the data.

As for the search of the parameters, there is also a Bayesian formulation of this problem, by having the different models appear as different values of a variable Π:

$$
\left\{
\begin{array}{l}
\text{?Preliminary knowledge?}(\pi) \left\{
\begin{array}{l}
\Delta = {}_1X \wedge \ldots \wedge {}_PX \\
\textbf{Dc} \\
\mathbf{P}(\Delta \wedge \Pi \mid \pi') = \mathbf{P}(\Pi \mid \pi') \times \mathbf{P}(\Delta \mid \Pi \wedge \\
\textbf{Fo} \\
\mathbf{P}(\Delta \mid [\Pi = \pi_1] \wedge \pi') = \prod_{k=1}^{P} \mathbf{P}({}_kX \mid \pi_1 \wedge \\
\mathbf{P}(\Delta \mid [\Pi = \pi_2] \wedge \pi') = \prod_{k=1}^{P} \mathbf{P}({}_kX \mid \pi_2 \wedge
\end{array}
\right. \\
\text{Data } (\delta)
\end{array}
\right\} \text{Ds} \qquad (15.7)
$$

- π' is the preliminary knowledge common to both models. For instance, usually both models share at least the same pertinent variables.

The question to be answered is $\mathbf{P}(\Pi \mid \delta \wedge \pi)$: the relative probability of the 2 models. The answer to this question is given by:

$$\mathbf{P}(\Pi \mid \delta \wedge \pi) = \mathbf{P}(\Pi \mid \pi') \times \mathbf{P}(\delta \mid \Pi \wedge \pi') \qquad (15.8)$$

Comparing 2 models, one usually tries to compute their relative probabilities:

$$
\begin{aligned}
\frac{\mathbf{P}(\pi_2 \mid \delta \wedge \pi)}{} &= \frac{}{\mathbf{P}(\pi_2 \mid \pi') \times \mathbf{P}(\delta \mid \pi_2 \wedge \pi')} \\
&= \frac{\mathbf{P}(\pi_1 \mid \pi')}{\mathbf{P}(\pi_2 \mid \pi')} \times \frac{\prod_{k=1}^{P} \mathbf{P}({}_kX \mid \pi_1 \wedge \pi')}{\prod_{P} \mathbf{P}({}_kX \mid \pi_2 \wedge \pi')}
\end{aligned} \qquad (15.9)
$$

This expression is most often very easy and fast to compute.

15.1.3 How to find interesting decompositions and associated parametric forms?

Given a set of pertinent variables, given a class of possible decompositions and of possible parametric forms, and given a data set δ, we want to select the best model in the class of possible ones.

This is basically is the same problem than the preceeding one, but extended to a large number of models (all the possible models of the class). We want here to find the most probable value of Π with the probability of Π given as above by:

$$\mathbf{P}(\Pi \mid \delta \wedge \pi) = \mathbf{P}(\Pi \mid \pi') \times \mathbf{P}(\delta \mid \Pi \wedge \pi') \tag{15.10}$$

This problem may be very difficult to solve as the number of possible models may be huge.

15.1.4 How to find the pertinent variables to model a phenomenon?

Finally, the most difficult problem of the four is to search for the pertinent variables in a given class of possible variables. It is the most difficult simply because the size of the search space is even larger than in the previous problem.

It is possible to give a bayesian formulation of this fourth problem, but this formulation is far too complicated to be presented here.

Let us now review some known algorithms and method that deal with one or several of the four preceeding questions.

15.2 Expectation - Maximization (EM)

The EM algorithm (Expectation-Maximization) (see Dempster, Laird & Rubin, 1977) is a general-purpose algorithm used to answer the first question: How to identify (learn) the value of the free parameters?

It is used in a wide variety of situations best described as *incomplete-data* problems. The idea behind the EM algorithm is intuitive and natural and it has been formulated and applied to a variety of problems. The EM algorithm can be applied either in incomplete-data situations where data are missing, distributions are truncated, observations are censored or grouped, or in situation where the incompleteness of data is not natural or evident.

The basic idea of the EM algorithm is to associate with the given incomplete-data problem, a *complete-data* problem for which ML estimation is computationally more tractable. The complete-data problem may yield a closed-form solution to the maximum likelihood estimate (MLE). The EM algorithm consists in reformulating the problem in terms of this more easily solved complete-data problem. It establishes a relationship between the likelihoods of these two problems, and exploits the simpler MLE computation of the complete-data problem, during the M-step of the iterative computing algorithm.

Usually, the E-step consists in producing data for the complete-data problem, using the observed data set of the incomplete-data problem and the current value of the parameters, so that the simpler M-step computation can be applied to this completed data set. During the E-step, it is precisely the log-likelihood of the data which is produced, and, as it is partly based on unobservable data, it is replaced by its conditional expectation given the observed data. Starting from suitable initial parameters values, E and M steps are iteratively repeated until convergence.

15.2.1 EM and bayesian networks

In Bayesian networks (see section 3.1.1), the EM algorithm is used to answer the first question: how to identify (learn) the value of the free parameters? The searched parameters are the values of the probabilty tables associated with the vertices and edges of the network.

When some nodes are hidden, the EM algorithm is used to find a locally optimal MLE. In the E-step, we compute the expected values of hidden nodes, given observed data. This is done by using an inference algorithm (for instance JLO), and then we treat these expected values as though they were observed. The M step is carried out by using the expected values of the hidden nodes as if they were observed. The solution is to compute the frequency of the values for each node until the difference between the new distribution and the older one is less than an arbitrary threshold.

15.2.2 EM and Mixture Models

EM is the most common algorithm used to identify the parameters of Mixture Models (see section 3.1.3).

For instance, for Gaussain mixtures, it is used to find where the gaussian kernels should be set and the values of their covariance matrices.

Numerous variants of EM have been developed in the context of mixture models. A synthesis may be found in *The EM Algorithm and Extensions* by McLachlan (McLachlan & Krishnam, 1997).

15.2.3 EM and HMM: The Baum-Welch Algorithm

The learning of the models in HMM (see section 3.1.2) is performed by the Baum-Welch algorithm, a specialized version of the EM algorithm, using the maximum likelihood esti-

mation criteria that determines the best model's parameters according to the set of data.

Intuitively, this algorithm counts the number of occurrences of each transition between the states and the number of occurrences of each observation in a given state in the training data. Each count is weighted by the probability of the alignment (state, observation).

Since many state sequences may generate a given output sequence, the probability that a HMM λ generates a sequence $\{o^0, ..., o^t\}$ is given by the sum over all state sequences (i.e, the marginal density of output sequences).

To avoid the combinatorial explosion, a recursive computation similar to the Viterbi algorithm can be used to evaluate the above sum. The forward probability $\alpha^t(i)$ is :

$$\begin{aligned} &= \sum_{S^0 \wedge ... \wedge S^{t-1}} \mathbf{P}(S^0 \wedge ... \wedge S^{t-1} \wedge [S^t = i] \wedge o^0 \wedge ... \wedge o^t) \\ &= \sum \mathbf{P}(S^0) \times \mathbf{P}(o^0 \mid S^0) \times \prod^t [\mathbf{P}(S^i \mid S^{i-1}) \times \mathbf{P}(o^i \mid S^i)] \end{aligned} \tag{15.11}$$

This probability represents the probability of ending at time t in state i and generating output $\{o^0, ..., o^t\}$ using all possible state sequences in between.

The Markov assumption allows the recursive computation of the forward probability as :

$$\alpha^{t+1}(j) = \sum_{i=1}^{N} [\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j])] \tag{15.12}$$

This computation is similar to Viterbi decoding except that a summation is used instead of a maximum.

Another useful quantity is the backward function $\beta^t(i)$, defined as the probability of starting at time t in state i and generating output $\{o^{t+1}, ..., o^T\}$, using all possible state sequences in between:

$$\beta^t(i) = \mathbf{P}(o^{t+1} \wedge ... \wedge o^T \wedge [S^t = i]) \tag{15.13}$$

The Markov assumption allows the recursive computation of the backward probability as :

$$\beta^t(i) = \sum_{j=1}^{N} [\beta^{t+1}(j) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j])] \tag{15.14}$$

To describe the procedure for reestimation of HMM parameters, we first define $\xi^t(i,j)$ as the posterior probability that the stochastic process accomplishes the transition $[S^t = i] \rightarrow [S^{t+1} = j]$, assuming the whole sequence of observations:

$$\xi^t(i,j) = \mathbf{P}([S^t = i] \wedge [S^{t+1} = j] \mid o^0 \wedge \ldots \wedge o^T) \tag{15.15}$$

We deduce:

$$\begin{aligned} \xi^t(i,j) &= \frac{\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j]) \times \beta^{t+1}(j)}{\mathbf{P}(o^0 \wedge \ldots \wedge o^T)} \\ &= \frac{\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j]) \times \beta^{t+1}(j)}{\sum_{N} \sum_{N} [\alpha^t(i) \times \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) \times \mathbf{P}(o^{t+1} \mid [S^{t+1} = j]) \times \beta^{t+1}(j)]} \end{aligned} \tag{15.16}$$

Finally, we define $\gamma^t(i)$ as the posterior probability that the process is in the state i at time t:

$$\gamma^t(i) = \sum_{j=1}^{N} \xi^t(i,j) \tag{15.17}$$

Therefore, the maximum likelihood (ML) estimate of the parameters of the HMM is:

$$\begin{aligned} \mathbf{P}([S^{t+1} = j] \mid [S^t = i]) &= \frac{\sum_{t=0}^{T} \xi^t(i,j)}{\sum_{t=0}^{T} \gamma^t(i)} \\ \mathbf{P}([O^t = k] \mid [S^t = i]) &= \frac{\sum_{t=0}^{T} \gamma^t(i)}{\sum_{t=0}^{T} \gamma^t(i)} \end{aligned} \tag{15.18}$$

See Rabiner's paper for details (Rabiner & Juang, 1986).

15.3 Problem Oriented Models

In problem oriented models, answering the first question is usually very easy. Indeed, the variables, the decomposition and the parametric forms have been chosen with 3 main

desiderata in mind:

1. Build a valid model of the observed phenomenon.

2. Build a simple enough model so that inference is kept tractable.

3. *Build a model so that the free parameters can easily be identified.*

For instance, in the sensor fusion models (see section 3.2.1), the sensor models $\mathbf{P}(S_i \mid \Phi)$ are very often Gaussian distributions. In that case, given N couples of values $\{({}_0s_i, {}_0\Phi), ..., ({}_Ns_i, {}_N\Phi)\}$, correponding to N observations, it is trivial to compute the means and standard deviations for each gaussian.

More details on this may be found in Lebeltel's Ph.D. thesis (Lebeltel, 1999) and more elaborate learning methods of this kind in Diard's Masters and Ph.D. thesis (Diard & Lebeltel, 1999 & Diard, 2003).

15.4 Learning Structure of Bayesian Networks

When learning the structure of a Bayesian network, one tries to solve question 3. Indeed, the problem is to find the best BN decomposition (i.e. the best graph structure).

Two main techniques are used to deal with this problem:

- Greedy hill-climbing: this algorithm starts from one (or several) random network and then "hill-climbs", by choosing iteratively a better network among its neighbours.

- MCMC Model Composition: This algorithm uses a Metropolis-Hastings algorithm (see section 4.3.2) to search the models using the acceptance rate:

$$a = \frac{\mathbf{P}(\pi_i \mid \delta \wedge \pi)}{\mathbf{P}(\pi^t \mid \delta \wedge \pi)} \tag{15.19}$$

as in equation [50].

Starting points to this subject are Heckerman's tutorial (Heckerman, 1995) and Buntime's one (Buntime, 1996). More details on MCMC methods may be found in 2 other papers (Murphy, 2001 & Friedman & Koller, 2000).

15.5 Bayesian Evolution?

Let us finish with some long term and very exciting perspectives.

If we admit that living systems are doing Bayesian inference and learning, then there must exist some natural processes, which have been answering and are still answering our 4 questions.

1. How to identify (learn) the value of the free parameters?

2. How to compare different probabilistic models (specifications)?

3. How to find interesting decompositions and associated parametric forms?

4. How to find the pertinent variables to model a phenomenon?

A tantalising answer is to say that natural evolution provided living beings with both the pertinent variables and the adequate decomposition and parametric forms. The pertinent variables may have been obtained by selecting the sensors and actuators in order to supply vital information. The decomposition would correspond to the structure of the nervous system, which basically expresses dependencies and conditional independencies between variables. The parametric forms can be seen as the information processing units implemented by neurons and assembly of neurons.

Given this apparatus, correponding to the preliminary knowledge, each individual in its lifetime can answer the first question by experimenting and learning the values of the free parameters of its nervous system.

We plan, in BIBA, to explore this possibility, by using evolutionary techniques to answer questions 2, 3 and 4. Some preliminary work has been already done in GRAVIR on that subject.

To the best of our knowledge, only Zhang (see its publications at http://bi.snu.ac.kr/Publications/pub_ei.html#BEA) really tried to explore the same path. However, our bibliographical study on this subject is not completed yet and we may have missed important works. It will be pursued next months.

16

Frequently Asked Question and Frequently Argued Matter

16.1 APPLICATIONS OF BAYESIAN PROGRAMMING (WHAT

$$$

17

Bibliography

Aji S. M. and McEliece R. J. ; (2000) ; The Generalized Distributive Law ; *IEEE Trans. Information Theory*, Vol. 46, No. 2

Arnborg, S., Corneil, D. G., & Proskurowski, A.; (1987);. Complexity of finding embedding in a k-tree; *SIAM J. Alg. Disc. Meth.*, 8(2), 277–284.

Bachelard, G. ; (1938) *La formation de l'esprit scientifique : Contribution a une psychanalyse de la connaissance objective* ; Librairie philosophique J. Vrin, Paris, France

Barlow, H.; (2001); The exploitation of regularities in the environment by the brain; *Behvioural and Brain Science* (BBS); 24 (4): 602-607.

Bellot, D. & Bessière, P.; (2003); Approximate Discrete Probability Distribution Representation using a Multi-ResolutionBinary Tree; *Proc. International Conference on Tools for Artificial Intelligence* - ICTAI 2003, Sacramento, California, USA, November 3-5

Bernouilli, J. (1713) *Ars Conjectandi*

Cooper, G. ; (1990) ; The computational complexity of probabilistic inference using Bayesian belief networks ; *Artificial Intelligence*, Vol. 42, pp. 393-405
www-ksl.stanford.edu/KSL_Abstracts/KSL-90-34.html

Darwiche, A. and Provan, G. ; (1997) ; Query DAGs: A Practical Paradigm for Implementing Belief-Network Inference ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 6, pp. 147-176

Dayan, P. & Abbott, L.F.; (2001); *Theoretical Neuroscience: Computational and Mathematical Modeling of Neural Systems*; MIT Press

Dechter, R.; (1996); Bucket elimination: A unifying framework for probabilistic inference. In E. Horvits and F. Jensen (Ed.), *Proc. Twelthth Conf. on Uncertainty in Artificial Intelligence*, pp. 211–219 Portland, Oregon.

Dechter, R. and Rish, I.; (1997); A scheme for approximating probabilistic inference; in *Proceedings of Uncertainty in Artificial Intelligence* (UAI97), pages 132-141

Forney, G.D. ; (1973) ; The Viterbi Algorithm ; *IEEE Transactions*, Vol. 61, p. 268-278

Glimcher, P.; (2003); *Decisions, Uncertainty, and the Brain: The Science of Neuroeconomics*; MIT Press

Glymour, C.; (2001); *The Mind's Arrows: Bayes Nets and Graphical Causal Models in Psychology*; MIT Press

Jaynes, E.T. (2003) *Probability theory - The logic of science*; Cambridge University Press

Jordan, M. & Sejnowski, T.; (2001); *Graphical Models: Foundations of Neural Computation*; MIT Press

Kaminka, G., Veloso, M., Scheffer, S., Solito, C., Adobatti, R., Marshall, A., Scholer, A. & Tejada, S. (2002) GameBots: a flexible test bed for multiagent team research; *Communication of the ACM*, Volume 45, Issue 1, Pages 43-45

Knill, D. & Richards, W.; (1996); *Perception as Bayesian inference*; Cambridge University Press

Laplace S. (1812) *Théorie analytique des probabilités*; Edition Veuve Courcier, Paris, France

Lauritzen, S. & Spiegelhalter, D. ; (1988) ; Local computations with probabilities on graphical structures and their application to expert systems ; *Journal of the Royal Statistical Society B* ; Vol. 50, pp. 157-224

Lauritzen, S. L. ; (1996) ; *Graphical Models* ; Oxford University Press

Li, Z., & D'Ambrosio, B.; (1994); Efficient inference in Bayes networks as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1), 55–81.

Pearl, J. (1988) *Probabilistic reasoning in intelligent systems : Networks of plausible inference* ; Morgan Kaufmann Publishers ; San Mateo, California, USA

Poincaré, H (1902) *La science et l'hypothèse*; Originally published in 1902, éditions de la Bohème, Rueil-Malmaison, FranceRao, R., Olshausen, B. and Lewicki, M.; (2002); *Probabilistic Models of the Brain: Perception and Neural Function*; MIT Press

Shachter, R. D., D'Ambrosio, B. D., & Del Favero, B. D.; (1990); Symbolic probabilistic inference in belief networks. In *Proc. 8th National Conference on Artificial Intelligence*, pp. 126–131 Boston.MIT Press.

Spiegelhalter, D.J.; (1986); Probabilistic reasoning in predictive expert systems, in: L.N. Kanal and J.F. Lemmer (eds.), *Uncertainty in Artificial Intelligence*, North-Holland, Amsterdam, pp. 47 -- 67.

Woodcock, S. (2001) Game AI: The state of the industry; *Game Developer Magazin*e

Zhang, N. L., & Poole, D.; (1994); A simple approach to Bayesian network computations. In *Proc. of the Tenth Canadian Conference on Artificial Intelligence*, pp. 171–178.

Zhang, N.L. and Poole, D. ; (1996) ; Exploiting Causal Independence in Bayesian Network Inference ; *Journal of Artificial Intelligence Research* (JAIR), Vol. 5, pp. 301-328